IBM DB2 Universal Database

System Monitor Guide and Reference

Version 6

SC09-2849-00

IBM DB2 Universal Database

System Monitor Guide and Reference

Version 6

SC09-2849-00

Before using this information and the product it supports, be sure to read the general information under “Appendix F. Notices” on page 411.

Contents

About This Book

Your DB2 Database Manager is instrumented to gather data on its operation and performance. You can use this data to:

- Monitor database activities
- Assist in problem determination
- Analyze performance
- Help configure the system.

The DB2 DBMS function that collects this data is called the database system monitor. This book describes how to use the database system monitor.

Various tools allow users to exploit the strengths of the database system monitor with minimal explicit knowledge of its associated commands, APIs, or data formats. Some of these tools, for example the Control Center, are described briefly, but for detailed information you should refer to the *Administration Guide*.

Who Should Use This Book

This book is for any users who require an understanding of the operation of the DB2 database system monitor, including how to program to its interface.

It is intended for database administrators, system administrators, security administrators and system operators who are maintaining a database accessed by local or remote clients. It is also for software developers who are interested in building software tools that use the DB2 database system monitor to assist in these administrative functions.

How This Book is Structured

This book starts with a description of the database system monitor and then details the data that you can collect with it.

Chapter 1. Introducing the Database System Monitor, introduces the database system monitor and describes its capabilities.

Chapter 2. Using the Database System Monitor, describes the information that is available from the database system monitor: how to collect it and how to work with it.

Chapter 3. Database System Monitor Data Elements, provides details of the information elements that you can collect with the database system monitor.

Chapter 4. Event Monitor Output, describes event monitor output and is primarily for programmers who want to write applications that read records from an event monitor trace.

Chapter 5. Snapshot Monitor Output, describes snapshot monitor output and is primarily for programmers who want to write applications that read snapshot records.

Appendix A. Database System Monitor Interfaces, contains detailed descriptions of the commands, SQL statements, APIs, and tools that you can use with the database system monitor.

Appendix B. Logical Data Groupings, lists all the structures in the self-describing snapshot and event monitor data streams, and the data elements associated with each.

Appendix C. Parallel Edition Version 1.2 Users, is intended for DB2 Parallel Edition Version 1.2. users of database system monitor who are upgrading their system to DB2 Version 6.

Appendix D. DB2 Version 1 sqlestat Users, is intended for DB2 Version 1 sqlestat users.

Appendix E. How the DB2 Library Is Structured describes the DB2 library; including books and online help.

Appendix F. Notices contains notice and trademark information.

Conventions

You will find this book easier to use if you look for these conventions:

- **Boldface type** indicates an important item or concept
- *Italics type* indicates new terms, data elements, configuration parameters, or book titles.
- `Monospace type` indicates an example of text that is displayed on the screen or contained in a file. It is also used for examples of sample code and calculations that can be performed.
- UPPERCASE TYPE indicates a file name, command name, or acronym.

Text in examples can be black or a lighter type.

db2 commands and output associated with the database system monitor are in black type

other db2 commands used are in lighter type

Chapter 1. Introducing the Database System Monitor

This chapter gives you a brief overview of the database system monitor's capabilities. It also discusses the integral role that the database system monitor plays in monitoring database activity and performance.

If you want to get started quickly, read this chapter and "Chapter 2. Using the Database System Monitor" on page 3. The information in these two chapters, combined with the reference material in "Appendix A. Database System Monitor Interfaces" on page 299, provides the information required to use the database system monitor.

"Chapter 3. Database System Monitor Data Elements" on page 35 provides complete details on all the data available with the database system monitor.

Database System Monitor Capabilities

The capabilities of the database system monitor opens several possibilities:

- **Activity monitoring**

  For example, using the database system monitor you can obtain:

  – The list of database connections:
    - The status of each connection.
    - The SQL that each is executing.
    - The locks that each holds.

  – The tables being accessed and the number of rows read and written for each.

  You can also track the progression of a query or application using information, such as:

  – The cursors that are currently open for this application.

  – The number of rows read or CPU consumed (if available from the operating system) by this application.

  – How long each query has been running.

  – How long an application has been idle.

- **Problem determination**

  You can collect data to help diagnose the cause of poor system and application performance. For example:

  - By tracing deadlocks you can determine conflicts between applications that lead to poor overall system performance.
  - By looking at the amount of time applications spent waiting for locks and which application is holding these locks you can identify applications that fail to commit their transactions, a common cause of poor system performance.
- **Performance analysis**

  You can use the information available to analyze the performance of individual applications or SQL queries. For example, you can monitor for:
  - The CPU consumed by each individual statement or application.
  - The time it takes to run a statement.
  - The number of rows read and returned.
  - The use of database resources, such as buffer pool, prefetchers, and SQL cache.
  - The number of times a particular DB2 dynamic SQL package has been executed.

  These run-time metrics are useful in tuning queries for optimal utilization of your database resources. Modifying a query or certain system parameters can result in dramatic performance improvements. The impact of your modifications can be measured with the database system monitor.

  You can also track the usage of indexes and tables, and in a partitioned database, the progression of a query on each partition. Adding indices or repartitioning the data often results in significant performance improvements.

  Carrying out some these performance analysis tasks may also require input that is obtained from the operating system, such as system load or the amount of free storage, or from other DB2 tools such as the **db2 explain facility**. For example, the db2expln application lets you analyze the *access plan* generated by the SQL compiler, which can then be compared with the run-time information available from the database system monitor.
- **System configuration**

  You can assemble the information necessary to evaluate and tune the effectiveness of your database manager and database configuration.

You can use the database system monitor to help monitor, tune, and manage your databases whether they are local or remote.

Chapter 2. Using the Database System Monitor

This chapter describes the data that is available from the DB2 Version 6 database system monitor. It explains how you can either take a **snapshot** of this data, or request the database manager to log information when certain **events** take place.

It describes the types of snapshots that you can take, and how they can be taken using CLP (command line processor) commands or APIs (application programming interfaces). It details the types of event monitors that can be used for data collection, and how to collect that information using commands and tools that come with DB2.

Database Manager Maintains Operation and Performance Data

Built into the database manager is the ability to collect data about its operation and performance, and that of the applications using it. The database manager maintains information at the following levels:

- Database manager
- Database
- DCS database
- Application (database connection)
- DCS application
- Table
- Table space
- Buffer pool
- Transaction
- DCS transaction
- Statement
- DCS statement
- Subsection
- Dynamic SQL package

Collecting some of this data introduces some processing overhead. For example, in order to calculate the execution time of an SQL statement, the database manager must make a call to the operating system to obtain timestamps before and after statement execution. These types of system calls are generally expensive. In order to minimize the overhead involved in maintaining monitoring information, **monitor switches** control the collection of potentially expensive data by the database manager.

Monitor Switches Control Data Collected by the Database Manager

The database system monitor will always collect some basic information, but you can use the switches to govern the amount of expensive data collected. Monitor switches can be set:

- **Explicitly**, this is usually done using the UPDATE MONITOR SWITCHES command.

  You can also set these switches in the database manager configuration file if you want data collection to start from the moment the server is started.

  Switches can be changed without stopping the database management system. This dynamic updating of switches requires that the application doing the update must be explicitly attached to the instance for the updates to take effect.

  Note: Any existing snapshot applications will not be affected by a dynamic update. In order to pick up the new default values for the switches, a monitoring application must terminate and re-establish its connection

  Switches are explained in "Resetting Monitor Data" on page 25. For more information on configuration see the *dft_monswitches* configuration parameters in the *Administration Guide.*

  Updating the switches in the database manager configuration file will update the switches for all nodes in a partitioned database if the user is attached to it.

- **Implicitly**, when an event monitor is activated. Event monitors are explained in "Event Monitors" on page 12.

To see if your database manager is currently collecting any monitor data issue the command:

```
     db2 get database manager monitor switches
```

The resulting output indicates the database manager switch settings and the time that they were turned on.

```
          DBM System Monitor Information Collected

Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information                       (LOCK) = OFF
Sorting Information                    (SORT) = ON   04-18-1997 10:11:01.738400
SQL Statement Information         (STATEMENT) = OFF
Table Activity Information            (TABLE) = OFF
Unit of Work Information                (UOW) = OFF
```

In this example, in addition to collecting basic-level information, the database manager is collecting all information under control of the sort switch.

Accessing Monitor Data

There are two ways to access the monitor data collected by the database manager:

- **Snapshot monitoring**

  Taking a snapshot gives you information for a specific point in time. A snapshot is a picture of the current state of activity in the database manager for a particular object or group of objects.
- **Event monitors**

  You can request the database manager to automatically log monitor data to files or a named pipe when specific events occur. This allows you to collect information about transient events that are difficult to monitor through snapshots, such as deadlocks and transaction completions.

Snapshot Monitoring

The snapshot monitor provides two categories of information for each level being monitored:

- State

  This includes information such as:
  - the current status of the database
  - information on the current or most recent unit of work
  - the list of locks being held by an application
  - the status of an application
  - the current number of connections to a database
  - the most recent SQL statement performed by an application
  - run-time values for configurable system parameters.
- Counters

  These accumulate counts for activities from the time monitoring started until the time a snapshot is taken. Such as:
  - the number of deadlocks that have occurred
  - the number of transactions performed on a database
  - the amount of time an application has waited on locks.

For example, you can obtain a list of the locks held by applications connected to a database by taking a database lock snapshot. First, turn on the LOCK switch (UPDATE MONITOR SWITCHES), so that the time spent waiting for locks is collected.

```
db2 connect to sample
db2 update monitor switches using LOCK on
db2 +c list tables for all          # this command will require locks
                                    # on the database catalogs
db2 get snapshot for locks on sample
```

Note: You can create and populate the sample database by running `sqllib/misc/db2sampl`.

Issuing the GET SNAPSHOT command returns the following.

```
          Database Lock Snapshot

 Database name                              = SAMPLE
 Database path                              = /home/bourbon/bourbon/NODE...
 Input database alias                       = SAMPLE
 Locks held                                 = 5
 Applications currently connected           = 1
 Applications currently waiting on locks    = 0
 Snapshot timestamp                         = 03-17-1999 15:40:29.976539

 Application handle                         = 0
 Application ID                             = LOCAL.bourbon.970411143813
 Sequence number                            = 0001
 Application name                           = db2bp_32
 Authorization ID                           = BOURBON
 Application status                         = UOW Waiting
 Status change time                         = Not Collected
 Application code page                      = 850
 Locks held                                 = 5
 Total wait time (ms)                       = 0

List of Locks
 Lock Object Name    = 4
 Object Type         = Row
 Tablespace Name     = SYSCATSPACE           Granted
 Table Schema        = SYSIBM
 Table Name          = SYSTABLES
 Mode                = NS
 Status              = Granted
 Lock Escalation     = NO

 Lock Object Name    = 2
 Object Type         = Table
 Tablespace Name     = SYSCATSPACE           Granted
 Table Schema        = SYSIBM
 Table Name          = SYSTABLES
 Mode                = IS
 Status              = Granted
 Lock Escalation     = NO

 Lock Object Name    = 259
 Object Type         = Row
 Tablespace Name     = SYSCATSPACE           Granted
 Table Schema        = SYSIBM
 Table Name          = SYSTABLES
 Mode                = NS
 Status              = Granted
 Lock Escalation     = NO

 Lock Object Name    = 7
 Object Type         = Table
 Tablespace Name     = SYSCATSPACE           Granted
 Table Schema        = SYSIBM
 Table Name          = SYSTABLES
 Mode                = IS
 Status              = Granted
 Lock Escalation     = NO

 Lock Object Name    = 0
 Object Type         = Internal P Lock
 Tablespace Name     =
 Table Schema        =
 Table Name          =
 Mode                = S
 Status              = Granted
 Lock Escalation     = NO
```

From this snapshot, you can see that there is currently one application connected to the SAMPLE database, and it is holding five locks.

```
    Locks held                                 = 5
    Applications currently connected           = 1
```

Note that the time (`Status change time`) when the `Application status` became `UOW Waiting` is returned as `Not Collected`, because the UOW switch is OFF.

The lock snapshot also returns the total time spent waiting for locks (so far), by applications connected to this database.

```
    Total wait time (ms)                       = 0
```

This is an example of an accumulating counter. “Resetting Monitor Data” on page 25 explains how counters can be reset to zero.

Authority Required for Snapshot Monitoring

To perform any of the snapshot monitor tasks, you must have SYSMAINT, SYSCTRL, or SYSADM authority for the database manager instance that you wish to monitor.

Snapshot Monitor Interface

Snapshot monitoring is invoked using the following application programming interfaces (APIs):

db2GetSnapshot()
: take a snapshot

sqlmon()
: set or query monitor switch settings

sqlmonrset()
: reset system monitor counters

sqlmonsz()
: estimate the size of the data that would be returned for a particular invocation of db2GetSnapshot()

db2ConvMonStream()
: convert Version 6 self-describing data streams to pre-Version 6 fixed size data structures

The Command Line Processor (CLP) provides a convenient command-based front-end to the snapshot APIs. For example, the GET SNAPSHOT command invokes the db2GetSnapshot() API.

Note: Starting in Version 6, the sqlmonss() - Get Snapshot API is replaced by the db2GetSnapshot() API.

"Appendix A. Database System Monitor Interfaces" on page 299 contains detailed information on the commands and APIs associated with the database system monitor.

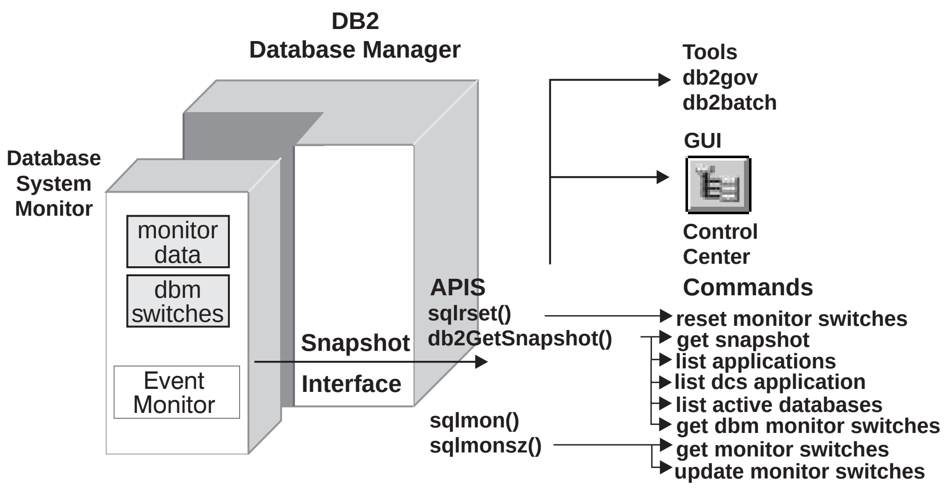


Figure 1. Snapshot Monitoring Interfaces

Information Available by Taking Snapshots

The following table lists all the supported snapshot request types. For certain request types, some information is returned only if the associated monitor switch is set ON. See "Chapter 3. Database System Monitor Data Elements" on page 35 to determine if a required counter is under switch control.

In the table, the API Request type column identifies the value that is supplied as input to the SQLMA input structure in the db2GetSnapshot() Snapshot API routine.

API request type	CLP command	Information returned
	List of connections	
SQLMA_APPLINFO_ALL	list applications [show detail]	Application identification information for all applications currently connected to a database that is managed by the DB2 instance on the node where snapshot is taken.
SQLMA_DBASE_APPLINFO	list applications for database *dbname* [show detail]	Same as SQLMA_APPLINFO_ALL for each application currently connected to the specified database.
SQLMA_DCS_APPLINFO_ALL	list dcs applications	Application identification information for all DCS applications currently connected to a database that is managed by the DB2 instance on the node where snapshot is taken.
	Database manager snapshot	

API request type	CLP command	Information returned
SQLMA_DB2	get snapshot for dbm	Database manager level information, including internal monitor switch settings.
	get dbm monitor switches	Returns internal monitor switch settings.
	Database snapshot	
SQLMA_DBASE	get snapshot for database on *dbname*	Database level information and counters for a database. Information is returned only if there is at least one application connected to the database.
SQLMA_DBASE_ALL	get snapshot for all databases	Same as SQLMA_DBASE for each database active on the node.
	list active databases	The number of connections to each active database. Includes databases that were started using the ACTIVATE DATABASE command, but have no connections.
SQLMA_DCS_DBASE	get snapshot for dcs database on *dbname*	Database level information and counters for a specific DCS database. Information is returned only if there is at least one application connected to the database.
SQLMA_DCS_DBASE_ALL	get snapshot for all databases	Same as SQLMA_DCS_DBASE for each database active on the node.
	Application snapshot	
SQLMA_APPL	get snapshot for application applid *appl-id*	Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
SQLMA_AGENT_ID	get snapshot for application agentid *appl-handle*	Same as SQLMA_APPL.
SQLMA_DBASE_APPLS	get snapshot for applications on *dbname*	Same as SQLMA_APPL, for each application that is connected to the database on the node.
SQLMA_APPL_ALL	get snapshot for all applications	Same as SQLMA_APPL, for each application that is active on the node.
SQLMA_DCS_APPL	get snapshot for dcs application applid *appl-id*	Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
SQLMA_DCS_APPL_ALL	get snapshot for all dcs applications	Same as SQLMA_DCS_APPL, for each DCS application that is active on the node.
SQLMA_DCS_APPL_HANDLE	get snapshot for dcs application agentid *appl-handle*	Same as SQLMA_DCS_APPL.
SQLMA_DCS__DBASE_APPLS	get snapshot for dcs applications on *dbname*	Same as SQLMA_DCS_APPL, for each DCS application that is connected to the database on the node.
	Table snapshot	

API request type	CLP command	Information returned
SQLMA_DBASE_TABLES	get snapshot for tables on *dbname*	Table activity information at the database and application level for each application connected to the database, and at the table level for each table that **was accessed** by an application connected to the database. Requires the table switch.
	Lock snapshot	
SQLMA_APPL_LOCKS	get snapshot for locks for application applid *appl-id*	List of locks held by the application. Also, lock wait information if any and the lock switch is ON.
SQLMA_APPL_LOCKS_AGENT_ID	get snapshot for locks for application agentid *appl-handle*	Same as SQLMA_APPL_LOCKS.
SQLMA_DBASE_LOCKS	get snapshot for locks on *dbname*	Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.
	Table space snapshot	
SQLMA_DBASE_TABLESPACES	get snapshot for tablespace on *dbname*	Information about table space activity at the database level, the application level for each application connected to the database, and the table space level for each table space that has been accessed by an application connected to the database. Requires the buffer pool switch.
	Buffer pool snapshot	
SQLMA_BUFFERPOOLS_ALL	get snapshot for all bufferpools	Buffer pool activity counters. Requires the buffer pool switch.
SQLMA_DBASE_BUFFERPOOLS	get snapshot for bufferpools on *dbname*	Same as SQLMA_BUFFERPOOLS_ALL, but for specified database only.
	Dynamic SQL snapshot	
SQLMA_DYNAMIC_SQL	get snapshot for dynamic sql on *dbname*	Point-in-time statement information from the SQL statement cache for the database.

Snapshot Uses an Instance Attachment or a Database Connection

Snapshot monitoring requires an instance attachment or a database connection. (An instance attachment is a connection between an application and an instance of the DB2 database manager.)

If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created. The instance attachment is usually done implicitly to the instance specified by the DB2INSTANCE environment variable when the first database system monitor API is invoked by the application. It can also be done explicitly, using the ATTACH TO command.

If there is both an instance attachment and a database connection, the instance attachment is used.

Once an application is attached or connected, all system monitor requests that it invokes are directed to that instance. This allows a client to monitor a remote server, by simply attaching to the instance, or connecting to one of the databases on it.

Dynamic SQL Snapshot

The DB2 statement cache stores packages and statistics for frequently used SQL statements. You can examine SQL activity by taking a snapshot of this cache. Due to the volume of records that can be returned from such a snapshot, a table function exists to view its contents (see "SQLCACHE_SNAPSHOT" on page 353).

A snapshot of the statement cache can only be taken over a database connection:

- GET SNAPSHOT FOR DYNAMIC SQL ON *database-alias* WRITE TO FILE command
- db2GetSnapshot API with a request type of SQLMA_DYNAMIC_SQL and iStoreResult set to TRUE

If write to file is attempted over an instance attachment, the request will be rejected.

Availability of Snapshot Monitor Data

If all applications disconnect from a database and the database deactivates, then the system monitor data for that database is no longer available. To obtain monitor information for all database activity during a given period you may want to use an event monitor. Alternatively, you can keep the database active until your final snapshot has been taken, either by starting it with the ACTIVATE DATABASE command, or by maintaining a permanent connection to the database.

Event Monitors

In contrast to taking a point in time snapshot, an event monitor writes out database system monitor data to either a file or a named pipe, when one of the following events occurs:

- end of a transaction
- end of a statement
- a deadlock
- start of a connection

- end of a connection
- database activation
- database deactivation
- end of a statement’s subsection (when a database is partitioned)
- flush event monitor statement issued.

An event monitor effectively provides the ability to obtain a **trace** of the activity on a database.

For example, you can request that DB2 logs the occurrence of deadlocks between connections to a database. First, you must create and activate a DEADLOCK event monitor:

For UNIX systems

Monitor Session

```
db2 connect to sample
db2 "create event monitor dlockmon for
  deadlocks write to file '/tmp/dlocks'"
mkdir /tmp/dlocks
db2 "set event monitor dlockmon state 1"
```

For OS/2 and Windows systems

Monitor Session

```
db2 connect to sample
db2 "create event monitor dlockmon for
  deadlocks write to file 'c:\tmp\dlocks'"
mkdir c:\tmp\dlocks
db2 "set event monitor dlockmon state 1"
```

Now, two applications using the database enter a deadlock. That is, each one is holding a lock that the other one needs in order to continue processing. The deadlock is eventually detected and resolved by the DB2 deadlock detector component, which will rollback one of the transactions. The following figures illustrate this scenario.

Application 1

```
db2 connect to sample
db2 +c "insert into staff values (1, 'Ofer',
  1, 'Mgr', 0, 0, 0)"
DB20000I  The SQL command completed
successfully.
```

Note: The +c option turns autocommit off for CLP.

Application 1 is now holding an exclusive lock on a row of the staff table.

Application 2

```
db2 connect to sample
db2 +c "insert into department values ('1',
  'System Monitor', '1', 'A00', NULL)"
DB20000I  The SQL command completed
successfully.
```

Application 2 now has an exclusive lock on a row of the department table.

Application 1

```
db2 +c select deptname from department
```

Assuming cursor stability, Application 1 needs a share lock on each row of the department table as the rows are fetched, but a lock on the last row cannot be obtained because Application 2 has an exclusive lock on it. Application 1 enters a LOCK WAIT state, while it waits for the lock to be released.

Application 2

```
db2 +c select name from staff
```

Application 2 also enters a LOCK WAIT state, while waiting for Application 1 to release its exclusive lock on the last row of the staff table.

These applications are now in a deadlock. This waiting will never be resolved because each application is holding a resource that the other one needs to continue. Eventually, the deadlock detector checks for deadlocks (see the *dlchktime* database manager configuration parameter in the *Administration Guide*) and chooses a victim to rollback:

Application 2

```
SQLN0991N  The current transaction has been
rolled back because of a deadlock or timeout.
Reason code "2". SQLSTATE=40001
```

At this point the event monitor logs a deadlock event to its target. Application 1 can now continue:

Application 1

```
DEPTNAME
-----------------------------
PLANNING
INFORMATION CENTER
 . . .
SOFTWARE  SUPPORT
SYSTEM MONITOR

   9 record(s) selected
```

Because an event monitor buffers its output and this scenario did not generate enough event records to fill a buffer, the event monitor values are forced to the event monitor output writer:

Monitor Session

```
db2 "flush event monitor dlockmon buffer"
DB20000I  The SQL command completed
successfully.
```

The event trace is written as a binary file. It that can now be formatted using the db2evmon tool:

Monitor Session

```
db2evmon -path /tmp/dlocks

Reading /tmp/dlocks/00000000.evt . . .
```

This will format and print to *stdout*, a trace similar to the following:

```
--------------------------------------------------------------------------
                            EVENT LOG HEADER
  Event Monitor name: DLOCKMON
  Server Product ID: SQL05000
  Version of event monitor data: 6
  Byte order: BIG ENDIAN
  Number of nodes in db2 instance: 1
  Codepage of database: 850
  Country code of database: 1
  Server instance name: bourbon
--------------------------------------------------------------------------

--------------------------------------------------------------------------
  Database Name: SAMPLE
  Database Path: /home/bourbon/bourbon/NODE0000/SQL00002/
  First connection timestamp: 06-03-1997 13:31:13.607548
  Event Monitor Start time:   06-03-1997 13:32:11.676071
--------------------------------------------------------------------------

3) Connection Header Event ...
  Appl Handle: 0
  Appl Id: *LOCAL.bourbon.970603173114                     - Monitor session
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.bourbon.970603173113
  Program Name    : db2bp_32
  Authorization Id: BOURBON
  Execution Id    : bourbon
  Codepage Id: 850
  Country code: 1
  Client Process Id: 63590
  Client Database Alias: sample
  Client Product Id: SQL05000
  Client Platform: AIX
  Client Communication Protocol: Local
  Client Network Name:
  Connect timestamp: 06-03-1997 13:31:13.607548

4) Connection Header Event ...
  Appl Handle: 1                                          - Application 1
  Appl Id: *LOCAL.bourbon.970603173330
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.bourbon.970603173329
  Program Name    : db2bp_32
  Authorization Id: BOURBON
  Execution Id    : bourbon
  Codepage Id: 850
  Country code: 1
  Client Process Id: 119710
  Client Database Alias: sample
  Client Product Id: SQL05000
  Client Platform: AIX
  Client Communication Protocol: Local
  Client Network Name:
  Connect timestamp: 06-03-1997 13:33:29.518568
```

```
5) Connection Header Event ...
  Appl Handle: 2
  Appl Id: *LOCAL.bourbon.970603173409                   - Application 2
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.bourbon.970603173408
  Program Name    : db2bp_32
  Authorization Id: BOURBON
  Execution Id    : bourbon
  Codepage Id: 850
  Country code: 1
  Client Process Id: 33984
  Client Database Alias: sample
  Client Product Id: SQL05000
  Client Platform: AIX
  Client Communication Protocol: Local
  Client Network Name:
  Connect timestamp: 06-03-1997 13:34:08.972643

6) Deadlock Event ...
  Number of applications deadlocked: 2                  - Deadlock
  Deadlock detection time: 06-03-1997 13:36:48.817786
  Rolled back Appl Id: : *LOCAL.bourbon.970603173409
  Rolled back Appl seq number: : 0001

7) Deadlocked Connection ...
  Appl Id: *LOCAL.bourbon.970603173409
  Appl Seq number: 0001
  Appl Id of connection holding the lock: *LOCAL.bourbon.970603173330
  Seq. no. of connection holding the lock:
  Lock wait start time: 06-03-1997 13:36:43.251687
  Deadlock detection time: 06-03-1997 13:36:48.817786
  Table of lock waited on      : STAFF
  Schema of lock waited on     : BOURBON
  Tablespace of lock waited on : USERSPACE1
  Type of lock: Row
  Mode of lock: X
  Lock object name: 39

8) Deadlocked Connection ...
  Appl Id: *LOCAL.bourbon.970603173330
  Appl Seq number: 0001
  Appl Id of connection holding the lock: *LOCAL.bourbon.970603173409
  Seq. no. of connection holding the lock:
  Lock wait start time: 06-03-1997 13:35:32.227521
  Deadlock detection time: 06-03-1997 13:36:48.817786
  Table of lock waited on      : DEPARTMENT
  Schema of lock waited on     : BOURBON
  Tablespace of lock waited on : USERSPACE1
  Type of lock: Row
  Mode of lock: X
  Lock object name: 15
```

This event monitor trace shows that there was 1 application connected to the database when the event monitor was activated. This is indicated by the first

Connection Header Event record in the output (record number 3). A Connection Event Header is generated for each active connection when an event monitor is turned on, and for each subsequent connection, once it becomes active. The other two Connection Headers, (records 4 and 5) were generated when the two applications connected.

The trace also shows that a deadlock occurred (record number 6). It shows which locks on which tables caused this deadlock (record numbers 7 and 8), and which application the deadlock detector chose to roll back (record number 6).

The **db2eva** graphical tool can also be used for formatting a trace. It is particularly useful for handling file traces that are too large to be read with db2evmon. It displays collected information in a tabular format. It includes a number of different view options, which allows you to filter unwanted records and drill down to the periods of interest in the trace. For instance, you can decide to display only the transaction events for a given connection. It also allows you to view the statement text for static SQL that it automatically fetches from the DB catalog (the text is only available for dynamic SQL in the event monitor trace).

You can invoke this tool with the db2eva command (see the *Command Reference*).

Note: The files must be available on the machine where you invoked db2eva.

The db2eva tool is available on OS/2 and Windows systems.

Authority Required for Event Monitoring

To define and use an event monitor on a database, you must have at least DBADM authority on that database.

Using Event Monitors

As illustrated in the sample scenario, collecting system monitor data with an event monitor is a three step process:

1. Create the event monitor
2. Activate the event monitor
3. Read the trace produced.

Create the event monitor.

Specify the events to be monitored. An event monitor is created and activated by using SQL statements. Unlike snapshot monitoring, where data can be collected at the database manager level, an event monitor only gathers data for a single database.

Creating an event monitor stores its definition in the event monitor database system catalogs:

SYSCAT.EVENTMONITORS	event monitors defined for the database
SYSCAT.EVENTS	events monitored for the database

It is necessary to connect to the database when defining an event monitor.

Activate the event monitor.

Activating an event monitor starts a process or thread, which records monitor data to either a named pipe or a file as events occur. You may want an event monitor to be activated as soon as the database is started, so that all activity is monitored from start-up. This can be done by creating an AUTOSTART event monitor:

```
db2 "create event monitor DLOCKMON
  for deadlocks write to file '/tmp/dlocks'
  AUTOSTART"
```

This event monitor will be automatically started every time the database is activated, either by using the ACTIVATE DATABASE command, or when the first application connects. Note that creating an AUTOSTART event monitor does not activate it. This event monitor will be activated the next time the database is stopped and re-activated. An event monitor that has not been automatically started must be manually started:

```
db2 set event monitor dlockmon state 1
```

All event monitors for a database are stopped when the database is deactivated.

Read the trace produced.

Reading a trace can be done using the **db2evmon** applet, or by writing your own application (see “Chapter 4. Event Monitor Output” on page 271). The

Control Center and Event Analyzer (parts of the DB2 GUI) can be used to create and activate event monitors, and to read the traces produced by FILE event monitors.

Figure 2 illustrates the process and interface for using event monitors.

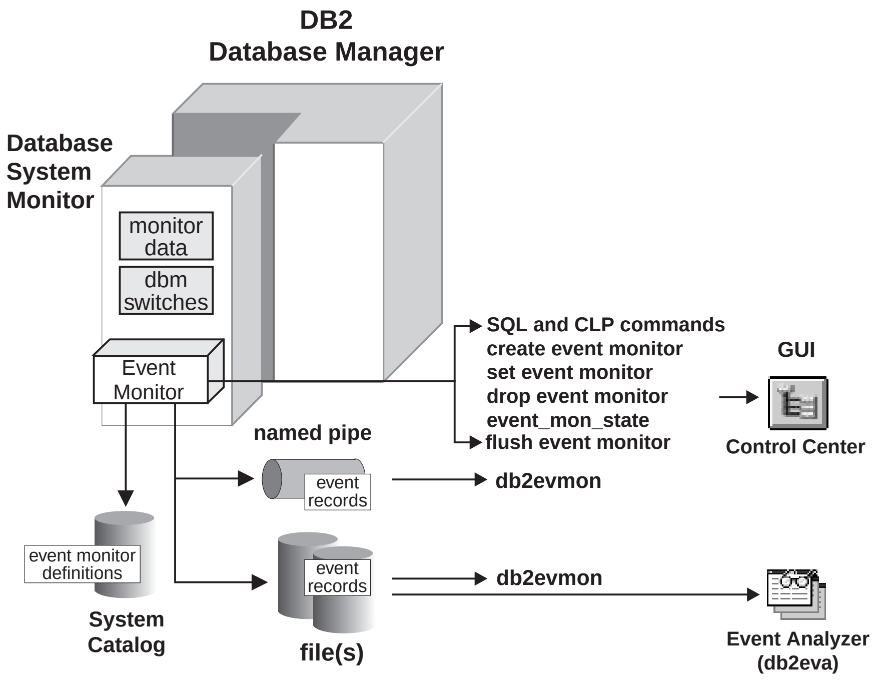


Figure 2. Event Monitoring Interfaces

As illustrated in Figure 2, event monitors are created and manipulated using the following SQL statements:

- CREATE EVENT MONITOR stores the event monitor definition in the database system catalogs for event monitors.
- SET EVENT MONITOR activates the event monitor, starting an **output thread** that will WRITE monitor data to either a file or named pipe. The trace produced can be formatted by the *db2evmon* or *db2eva* tools.
- DROP EVENT MONITOR deletes the event monitor definition from the database system catalogs for event monitors. An active event monitor cannot be dropped.
- FLUSH EVENT MONITOR forces monitor values to the event monitor output writer.

Querying the State of an Event Monitor

You can determine if an event monitor is active by using the SQL function EVENT_MON_STATE:

```
db2 connect to sample
db2 "select evmonname, EVENT_MON_STATE(evmonname)
from syscat.eventmonitors"

NAME            2
--------------  -------
DLOCKMON        0

  1 record(s) selected
```

A returned value of 0 indicates that the event monitor is inactive.

Information Available from Event Monitors

Event monitors return information that is similar to the information available using the snapshot API. In these cases, it is an event that controls when the snapshot is taken. For example, a connection event monitor basically takes an application snapshot just before the connection is terminated.

Event Types

When you define an event monitor you must declare the event types that will be monitored. The following table lists the event types supported and indicates the information returned. **Note:** an event monitor can be defined for more than one event type.

Event type	When data is collected	Information returned
Deadlock	Detection of a deadlock	The applications involved and locks in contention.
Statements	End of SQL statement	Statement start/stop time, CPU used, text of dynamic SQL, SQLCA (return code of SQL statement), and other metrics such as fetch count.
	End of subsection	For partitioned databases: CPU consumed, execution time, table and tablequeue information.
Transactions	End of unit of work	Unit of work start/stop time, previous UOW time, CPU consumed, locking and logging metrics.
Connections	End of connection	All application level counters.

Event type	When data is collected	Information returned
Database	Database deactivation or last connect reset	All database level counters.
Buffer pools		Counters for buffer pool, prefetchers, page cleaners and direct I/O for each buffer pool.
Table spaces		Counters for buffer pool, prefetchers, page cleaners and direct I/O for each table space.
Tables		Rows read/written for each table.

Note: In addition to the above information, all event monitors trace the establishment of connections to the database, by generating a *connection header record* for each active connection when the event monitor is turned ON, and for each subsequent connection, thereafter.

See "Output Records" on page 271 for a list of the records generated for each event type.

Using Pipe Event Monitors

A pipe event monitor allows you to process event records in real time. Another important advantage to using pipe event monitors is that your application can ignore unwanted data as it reads it off the pipe, giving the opportunity to considerably reduce storage requirements. It also allows an application to store event monitor data, in real-time, into an SQL database.

When you direct data to a pipe, I/O is always blocked and the only buffering is that performed by the pipe. It is the responsibility of the monitoring application to promptly read the data from the pipe as the event monitor writes the event data. If the event monitor is unable to write the data to the pipe (for example, because the pipe is full), monitor data will be lost.

The steps for using pipe event monitors are essentially the same on all operating systems. However, implementation can be different. The following section describes the basic steps, and highlights the differences between UNIX based systems, Windows NT, and OS/2.

1. Define the event monitor

```
  db2 connect to sample
On AIX, and other UNIX platforms:
  db2 create event monitor STMT2 for statements
      write to PIPE '/tmp/evmpipe1'
On Windows NT:
  db2 create event monitor STMT2 for statements
      write to PIPE '\\.\pipe\evmpipe1'
```

```
On OS/2:
  db2 create event monitor STMT2 for statements
      write to PIPE '\pipe\evmpipe1'
```

2. Create the named pipe

   In UNIX (this includes AIX environments), use the mkfifo() function or mkfifo command. In OS/2, use the DosCreateNPipe() function. In Windows NT, use the CreateNamedPipe() function. The pipe name must be the same as the target path specified on the CREATE EVENT MONITOR statement.

3. Open the named pipe

   In UNIX, use the open() function. In OS/2, use the DosConnectNPipe() function. In Windows NT, use the ConnectNamedPipe() function.

   You can also use the db2evmon application, specifying the database and pipe name, for example:

   ```
   db2evmon -db sample -evm STMT2
   ```

   This will open the named pipe and wait for the event monitor to write to it.

4. Activate the event monitor

   If the event monitor is started automatically, you do not need to take any specific action to start it unless the database is already active (however, the pipe must already be opened).

   ```
   db2 set event monitor stmt2 state 1
   ```

5. Read data from the named pipe

   In UNIX, use the read() function. In OS/2, use the DosRead() function. In Windows NT, use the ReadFile() function. Your application may stop reading data from the pipe at any time. When it reads an EOF, there is no more monitor data. See "Chapter 4. Event Monitor Output" on page 271 for how to read the event monitor data.

6. Deactivate the event monitor

   ```
   db2 set event monitor stmt2 state 0
   ```

   This statement can be used to stop any event monitor, even one that was started automatically. If you do not explicitly stop an event monitor, it will be stopped when:

   - The last application disconnects from the database
   - It experiences an error while writing to the named pipe: for example, the monitoring application closes the pipe before deactivating the event monitor. In this case, the event monitor will turn itself off and log a system-error-level message in the diagnostic log, db2diag.log.

7. Close the named pipe.

In UNIX, use the close() function. In OS/2, use the DosDisConnectNPipe() function. In Windows NT, use the DisconnectNamedPipe() function.

8. Delete the named pipe.

   In UNIX, use the unlink() function. In OS/2, use the DosClose() function. In Windows NT, use the CloseHandle() function.

For UNIX-based operating systems, named pipes are like files, so you do not have to delete them and create them again before each use.

Pipe Overflows

In addition, there must be enough space in the named pipe. If the application does not read the data fast enough from the named pipe, the pipe will fill up and overflow. Pipe overflows can also occur on platforms (such as OS/2) where the creator of the pipe can define the size of the named pipe buffer. The smaller the buffer, the greater the chance of an overflow occurring. When a pipe overflow occurs, the monitor creates overflow event records indicating that an overflow has occurred. The event monitor is not turned off, but monitor data is lost. If there are outstanding overflow event records when the monitor is deactivated, a diagnostic message will be logged. Otherwise, the overflow event records will be written to the pipe when possible.

If your operating system allows you to define the size of the pipe buffer, use a pipe buffer of at least 32K. For high-volume event monitors, you should set the monitoring application's process priority equal to or higher (lower nice value on AIX) than the agent process priority (see the section on Priority of Agents in the *Administration Guide*).

When Counters are Initialized

The data collected by the database manager includes several accumulating counters. These counters are incremented during the operation of the database, for example, every time an application commits a transaction.

Counters are initialized when their applicable object becomes active. For example, the number of buffer pool pages read for a database (a basic element) is set to zero when the database is activated.

Counters under switch control are reset to zero when their associated switch is turned on.

Counters returned by event monitors are reset to zero when the event monitor is activated.

Resetting Monitor Data

Each event monitor and any application using the snapshot monitor APIs has its own logical view of the DB2 monitor data and switches. This means that when counters are reset or initialized, it only affects the event monitor or application that reset or initialized them.

Event monitor data cannot be reset, except by turning the event monitor off, and then on again.

An application taking snapshots can reset its view of the counters at any time by using the RESET MONITOR command.

When issuing its first snapshot API, an application inherits the default settings from the database manager configuration. For example, assuming that the statement switch was set in the database manager configuration file:

```
db2 update dbm cfg using DFT_MON_STMT on
db2start
```

Issuing a GET MONITOR SWITCHES command

```
db2 get monitor switches
```

will show that the statement switch is ON.

```
          Monitor Recording Switches

Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information                       (LOCK) = OFF
Sorting Information                    (SORT) = OFF
SQL Statement Information         (STATEMENT) = ON   05-25-1997 10:44:34.820446
Table Activity Information            (TABLE) = OFF
Unit of Work Information                (UOW) = OFF
```

Turning OFF the statement switch from the command line will only affect the application issuing the command. The statement switch will still be ON for other applications (unless they have also turned it OFF). For example:

```
db2 update monitor switches using STATEMENT OFF
DB20000I The UPDATE MONITOR SWITCHES command completed successfully
```

Then query your application’s switches.

```
db2 get monitor switches
```

```
          Monitor Recording Switches

Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information                       (LOCK) = OFF
Sorting Information                    (SORT) = OFF
SQL Statement Information         (STATEMENT) = OFF
Table Activity Information            (TABLE) = OFF
Unit of Work Information                (UOW) = OFF
```

Querying the database manager switches will show that the update did not affect its settings:

```
db2 get database manager monitor switches
```

```
          DBM System Monitor Information Collected

Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information                       (LOCK) = OFF
Sorting Information                    (SORT) = OFF
SQL Statement Information         (STATEMENT) = ON     05-25-1997 10:44:34
Table Activity Information            (TABLE) = OFF
Unit of Work Information                (UOW) = OFF
```

When a monitoring application turns off a monitor switch or resets a data element counter, the DB2 server does not reset its own internal counters. Instead, it re-initialize the private **logical view** for that user. Other monitoring applications or event monitors are not affected.

You must have SYSADM, SYSCTRL, or SYSMAINT authority to use the UPDATE MONITOR SWITCHES command. See *Command Reference* for information on this command.

The database manager keeps track of all the applications using the snapshot monitor APIs and their switch settings. If a switch is set in its configuration, then the database manager always collects that monitor data. If a switch is OFF in the configuration, then the database manager will collect data as long as there is at least one application with this switch turned ON.

Internally, event monitors also use switches to instruct the engine as to which data should be collected. However, this is an implementation issue, and the switch settings for a particular event monitor cannot be queried.

An actual DBMS monitor switch is set as long as at least one application or event monitor needs it, or if it is set in the configuration file.

System Monitor Memory Requirements - (mon_heap_sz)

The memory required for maintaining the private views of the database system monitor system monitor data is allocated from the monitor heap. Its size is controlled by the *mon_heap_sz* configuration parameter. The amount of memory required for monitoring activity varies widely depending on the number of monitoring applications and event monitors, the switches set, and the level of database activity. The following formula provides an approximation of the number of pages required for the monitor heap.

```
(number of monitoring applications + 1) *
(number of databases *
  (800 + (number of tables accessed * 20) +
    ((number of applications connected + 1) *
      (600 + (number of table spaces * 100)))))
/ 4096
```

You may need to experiment with this value, increasing it if monitor commands occasionally fail with an SQLCODE of -973, when the database manager switches are on.

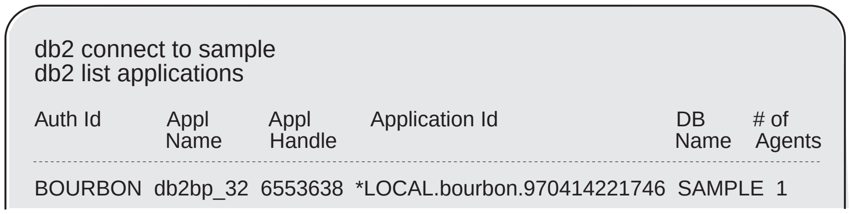
Partitioned Database Considerations

The database system monitor interface is the same for all types of systems, whether they use single partition or multiple partition databases and whether intra-query parallelism is used. All the commands and APIs are exactly the same. The only difference is the output; more complex systems generally return more information.

Taking a Snapshot on Multi-node Systems

On systems that use inter-partition parallelism, taking a snapshot only returns monitor data from the instance where the application is attached. For example, assuming a table that is located in two database partitions, that is some of its rows are stored on one node (Node 100) and others are stored on another node (Node 200).

Node 100



```
db2 connect to sample
db2 list applications

Auth Id    Appl      Appl      Application Id                  DB      # of
           Name      Handle                                    Name    Agents
-----------------------------------------------------------------------------
BOURBON  db2bp_32  6553638  *LOCAL.bourbon.970414221746  SAMPLE  1
```

Taking a snapshot on Node 200 initially returns no data:

Note: The LIST APPLICATION command uses the database system monitor. Invoking it actually calls the the snapshot API db2GetSnapshot() with a request of type SQLMA_APPLINFO_ALL.

Node 200

```
db2 list applications
SQL1611W  No data was returned from Database System Monitor.
```

Now, issuing a query from Node 100 will result in a secondary connection to Node 200 to fetch the rows that reside in that partition:

Node 100

```
db2 +c select lastname from employee

Huras
Ofer
Bourbonnais
Musker
Cartwright
```

Now there is a subagent for the application running on Node 200:

Node 200

```
db2 list applications

Auth Id     Appl       Appl       Application Id                     DB       # of
            Name       Handle                                        Name     Agents
-----------------------------------------------------------------------------------
BOURBON  db2bp_32  6553638  *LOCAL.bourbon.970414221746  SAMPLE  1
```

And there are now two agents running on Node 100; the coordinator agent and a subagent:

Node 100

```
db2 list applications

Auth Id     Appl       Appl       Application Id                     DB       # of
            Name       Handle                                        Name     Agents
-----------------------------------------------------------------------------------
BOURBON  db2bp_32  6553638  *LOCAL.bourbon.970414221746  SAMPLE  2
```

On the non-coordinating node, you can determine where the coordinator resides, and check if the application originated on the node that issued the snapshot, using:

Node 200

```
db2 list application show detail

. . . Appl       Application Id                 . . . Coordinating   Coordinator
      Handle                                           Node Number   pid/thread
      ----------------------------------------------------------------------------
      6553638  *LOCAL.bourbon.970414221746       100            66204
```

The *Application Handle* returned, 6553638 is unique across all nodes. The *node number* corresponds to one of the nodes listed in the *db2nodes.cfg* configuration file (see the *Administration Guide*).

Using the application handle, you can request monitor information on any node by issuing a GET SNAPSHOT FOR APPLICATION, which will return data if the application is connected on that node. You can also FORCE the application, which will work from any node:

Node 200

```
db2 force application (6553638)
DB20000I The FORCE APPLICATION command completed successfully.
DB221024I This command is asynchronous and may not be effective
immediately.
```

Using Event Monitors on Multi-node Systems

An event monitor uses an operating system process or a thread to write its trace. The node where this process or thread runs is called the **monitor node**. An event monitor can be monitoring events as they occur locally on the monitor node, or globally as they occur on any node where the DB2 database manager is running. A global event monitor writes a single trace that contains activity from all nodes.

Whether an event monitor is local or global is referred to as its **monitoring scope**. Both the monitor node and monitor scope are part of an event monitor's definition. For example:

```
db2 connect to sample
db2 "create event monitor DLOCKS for
  deadlocks write to file '/tmp/dlocks'
  ON NODE 5 GLOBAL"
```

This global event monitor will report deadlocks that involve any nodes in the system. Its I/O component will physically run on Node 5, writing its records to files in the /tmp/dlocks directory on that node.

You can look at the definition for this monitor in the system catalog:

```
db2 "select evmonname,nodenum, monscope
from syscat.eventmonitors"

EVMONNAME     NODENUM     MONSCOPE
------------------   ---------------   ----------------
DLOCKS              5              G

   1 record(s) selected
```

The returned information shows event monitor DLOCKS is defined as global and its monitor node is 5.

Note: Only deadlock event monitors can be defined as global, all other event monitors must be defined as local.

Monitoring Subsections

On systems that use inter-partition parallelism, the SQL compiler partitions the access plan for an SQL statement into **subsections**. Each subsection is executed by a different DB2 agent.

The access plan for an SQL statement generated by the DB2 code generator during compilation can be obtained using the **db2expln** or **dynexpln** commands (see the *Command Reference*). As an example, selecting all the rows from a table that is partitioned across several nodes might result in an access plan having two subsections:

1. Subsection 0, the coordinator subsection, whose role is to collect rows fetched by the other DB2 agents (subagents) and return them to the application.
2. Subsection 1, whose role is to perform a table scan and return the rows to the coordinating agent.

In this simple example, subsection 1 would be distributed across all the database partitions. There would be a subagent executing this subsection on each physical node of the **nodegroup** to which this table belongs. See *Administration Guide* for more information on these concepts.

The database system monitor allows you to correlate run-time information with the access plan, which is compile-time information. With inter-partition parallelism, it breaks information down to the subsection level. For example, when the statement monitor switch is ON, a GET SNAPSHOT FOR APPLICATION will return information for each subsection executing on this node, as well as totals for the statement.

The subsection information returned for an application snapshot includes:

- the number of table rows read / written
- CPU consumption
- elapsed time
- the number of tablequeue rows sent and received from other agents working on this statement. This allows you to track the execution of a long running query by taking a series of snapshots.
- subsection status. If the subsection is in a WAIT state, because it is waiting for another agent to send or receive data, then the information also identifies the node or nodes preventing the subsection from progressing in its execution. You may then take a snapshot on these nodes to investigate the situation.

The information logged by a statement event monitor for each subsection after it has finished executing includes: CPU consumption, total execution, time, and several other counters.

Monitor Output Format

Version 6 introduces a new output format for snapshot and event monitors. Rather than returning a list of data structures, the system monitor now returns a self-describing output data stream. The move to this new format coincides with DB2 Universal Database's added flexibility when it comes to naming SQL objects. For example, in Version 6 table names grew from a maximum of 18 bytes, to a maximum of 128 bytes. The static sized output structures used in previous releases could not contain this change in size.

This self-describing data stream allows you to parse through the returned data. It also means that any changes to existing data elements, or the addition of new data elements will not require changes to existing applications.

The returned monitor data is in the following format:

size	The size (in bytes) of the data stored in the data element or logical data grouping. In the case of a logical data grouping, this is the size of all data in the logical group (for example, the database logical grouping (*db*) contains individual data elements (for example, *total_log_used*) along with other logical data groupings, such as rollforward information (*rollforward*).
type	The type of element stored in the data (for example, variable length string or signed 32 bit numeric value). An element type of *header* refers to a logical data grouping for an element (see “Output Records” on page 271 and “Snapshot Requests” on page 287.)
element	The name of the data element that was captured by the monitor. In the case of a logical data grouping, this is the name of the group (for example, *collected*, *dbase*, or *db_event*).
data	The value collected by a monitor for a data element. In the case of a logical data grouping, there is no data section. Strings returned by DB2 are NOT NULL TERMINATED.

“Chapter 4. Event Monitor Output” on page 271 and “Chapter 5. Snapshot Monitor Output” on page 287 provide examples of the event monitor and snapshot data streams.

When a Version 6 snapshot request is made, but a lower version of snapshot data is returned from the server (for example, from a down-level server), SQLCODE +1627W is returned to the caller, and the monitor output is in the pre-Version 6 format and must be parsed using the Version 5 method (see Table 3 on page 293).

The db2ConvMonStream API can be used to convert the new monitor format for a logical data grouping to the corresponding pre-Version 6 data structure. “db2ConvMonStream” on page 310 describes this API and maps pre-Version 6 structures to the new Version 6 format. For more information on the data returned by Version 6 snapshot monitors see “Chapter 5. Snapshot Monitor Output” on page 287.

Event monitors write their data in the new monitor format by default. This can be overridden for individual event monitors by setting the registry variable *DB2OLDEVMON=evmon1,evmon2,...*, where *evmon1* is an event monitor that will write its data in the old format. For more information on the data returned by Version 6 event monitors see “Chapter 4. Event Monitor Output” on page 271.

DB2 Productivity Tools

The database system monitor is a very powerful function of the DB2 database manager. It can be exploited to develop productivity tools for the database administrator (DBA) and database developer. The following are a few examples of productivity tools that use the function of the database system monitor, and are included with the DB2 product:

- Control Center

  A GUI for performance and event monitoring. For performance, it allows you to define variables in terms of the metrics returned by the database system monitor and graph them over time. For example, you can request that it take a snapshot and graph the progression of a performance variable over the last eight hours. **Alerts** can be set to notify the DBA when certain threshold are reached. For event monitors, it allows you to create, activate, start, stop, and delete event monitors. See the online help for the Control Center for more information.
- db2batch

  An application that uses snapshot monitoring to collect metrics for tuning SQL queries. See the *Command Reference*and the *Administration Guide* for more information.
- db2gov

  The DB2 governor is an application that uses snapshot monitoring to supervise the load and usage of the database manager. It provides the functions to FORCE or change the run-time priority of applications exceeding certain limits. These limits are specified by the DBA in the db2gov configuration file. Application limits and privileges can be expressed using several different parameters, for example maximum amount of CPU. See the *Command Reference* and the *Administration Guide* for more information.
- db2evmon

  An application that formats the data stream created by an event monitor. See the *Command Reference* for more information.
- Control Center

  A GUI for snapshot and event monitoring. For snapshots, it allows you to define performance variables in terms of the metrics returned by the database system monitor and graph them over time. For example, you can request that it take a snapshot and graph the progression of a performance variable over the last eight hours. **Alerts** can be set to notify the DBA when certain threshold are reached. For event monitors, it allows you to create, activate, start, stop, and delete event monitors. See the online help for the Control Center for more information.
- Event Analyzer

A GUI for viewing file event monitor traces. Information collected on connections, deadlocks, overflows, transactions, statements, and subsections is organized and displayed in a tabular format. See the online help for the Event Analyzer for more information.

- Windows NT Performance Monitor

  System monitor counters for DB2 Universal Database and DB2 Connect have been added to the Windows NT Performance Monitor. See the Help for the Windows NT Performance Monitor for information on accessing database manager, database, DB2 Connect database, application, and DB2 Connect application counters.

Chapter 3. Database System Monitor Data Elements

This chapter describes the information that is available from the database system monitor. The information returned by database system monitor falls into the following categories:

- **Identification** for the database manager, an application, or a database connection being monitored.
- Data primarily intended to help you to **configure** the system.
- Database **activity** at various levels including database, application, table, or statement. This information can be used for activity monitoring, problem determination, and performance analysis. But it can also be used for configuration.
- Information on **DB2 Connect** applications. Including information on DCS applications running at the gateway, SQL statements being executed, and database connections.

In this chapter, data elements are organized by their primary use category. When applicable, elements that have multiple uses may be referred to by associated elements in other categories. Multi-use data element information only appears in its main category, it is not duplicated in other categories. Refer to *data elements* in the Index, if you have trouble finding a data element.

The information is grouped as follows:

- "Server Identification and Status" on page 38
- "Database Identification and Status" on page 44
- "Application Identification and Status" on page 50
  - "DB2 Agent Information" on page 77
- "Database Manager Configuration" on page 78
  - "Agents and Connections" on page 78
  - "Sort" on page 94
  - "Hash Join" on page 101
  - "Fast Communication Manager" on page 104
- "Database Configuration" on page 110
  - "Buffer Pool Activity" on page 110
    - "Extended Storage" on page 135
  - "Non-buffered I/O Activity" on page 140

  - “Catalog Cache” on page 146
  - “Package Cache” on page 150
  - “Database Heap” on page 156
  - “Logging” on page 157
- “Database and Application Activity” on page 164
  - “Locks and Deadlocks” on page 164
  - “Lock Wait Information” on page 176
  - “Rollforward Monitoring” on page 184
  - “Table Activity” on page 186
  - “SQL Cursors” on page 197
  - “SQL Statement Activity” on page 202
  - “SQL Statement Details” on page 215
  - “Subsection Details” on page 227
  - “Intra-query Parallelism” on page 237
  - “CPU Usage” on page 238
  - “Snapshot Monitoring Elements” on page 247
  - “Event Monitoring Elements” on page 250
- “DB2 Connect” on page 254
  - “Transaction Processor Monitoring” on page 267

Note: For Enterprise - Extended Edition users, snapshot elements only apply to the partition where the snapshot was issued.

How to Read the Data Element Tables

The section for each data element begins with a table that lists standard information. An example is shown in Figure 3, followed by an explanation of each part of the table.

① Snapshot Level	② Logical Data Grouping	③ Monitor Switch
Database	dbase	Sort
Application	appl	Sort

④ **Resettable**	Yes

⑤ Event Type	⑥ Logical Data Grouping
Database	db_event
Connection	conn_event
Statement	stmt_event

⑦ **Element Name**	total_sorts
Element Type	counter

⑧ **Related Information**	See Resettable See Switches See Sort Overflows

Figure 3. Sample Element Table

1. The level of information that can be captured by the snapshot monitor.
2. The data group where captured snapshot information is returned. If parsing the data stream directly, the element name is uppercased and prefixed with SQLM_ELM_.
3. The snapshot monitor switch that must be set to obtain this information.
4. Whether or not the counter can be reset (snapshot monitor only).
5. The event monitor must be created with this event type to collect this information. See “Element Types”.
6. The data group in which captured event information is returned. If parsing the data stream directly, the element name is uppercased and prefixed with SQLM_ELM_.
7. The name and type of element, as returned in the data group. If parsing the data stream directly, the element name is uppercased and prefixed with SQLM_ELM_.
8. References to related data elements or monitoring concepts.

This table is followed by a description of the element and information on how you can use it when monitoring your database.

Element Types

Data elements are classified by the following categories:

- Counter

A counter counts the number of times an activity occurs. Counter values increase during monitoring. Most are resettable.

- Gauge

  A gauge indicates the current value for an item. This value can go up and down depending on database activity (for example, the number of locks held).
- Water mark

  A water mark indicates the highest (maximum) or lowest (minimum) value an element reached since monitoring was started. These are not resettable.
- Information

  An information element provides reference-type details of your monitoring activities. This can include items such as node names, aliases, and path details.
- Timestamp

  A timestamp indicates the date and time that an activity took place, by providing the number of seconds and microseconds that have elapsed since January 1, 1970. In the C language, for example, this can be converted to calendar date and time using the `ctime()` function.
- Time

  Time returns the number of seconds and microseconds spent on an activity.

Server Identification and Status

The following elements provide identification and status information about the server:

- "Start Database Manager Timestamp" on page 39
- "Configuration NNAME at Monitoring (Server) Node" on page 39
- "Server Instance Name" on page 40
- "Database Manager Type at Monitored (Server) Node" on page 40
- "Server Product/Version ID" on page 41
- "Server Version" on page 41
- "Service Level" on page 42
- "Server Operating System" on page 42
- "Product Name" on page 43
- "Product Identification" on page 43
- "Status of DB2 Instance" on page 44
- "Time Zone Displacement" on page 44

Start Database Manager Timestamp

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Resettable	No	
Element Name	db2start_time	
Element Type	timestamp	
Related Information	• "Snapshot Time" on page 249	

Description: The date and time that the database manager was started using the `db2start` command.

Usage: This element may be used with the *Snapshot Time* monitor element to calculate the elapsed time since the database manager was started up until the snapshot was taken.

Configuration NNAME at Monitoring (Server) Node

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic
Resettable	No	
Element Name	server_nname	
Element Type	information	
Related Information	• "Configuration NNAME of Client" on page 60	

Description: The name of the node being monitored by the database system monitor.

Usage: This element can be used to identify the database server node you are monitoring. This information can be useful if you are saving your monitor output in a file or database for later analysis and you need to differentiate the data from different database server nodes. This node name is determined based on the *nname* configuration parameter.

Server Instance Name

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	collected	Basic
Resettable	No	

Event Type	**Logical Data Grouping**
Event Log Header	event_log_header
Element Name	server_instance_name
Element Type	information
Related Information	• "Configuration NNAME at Monitoring (Server) Node" on page 39

Description: The name of the database manager instance for which the snapshot was taken.

Usage: If more than one instance of the database manager is present on the same system, this data item is used to uniquely identify the instance for which the snapshot call was issued. Along with *Configuration NNAME at Monitoring (Server) Node,* this information can be useful if you are saving your monitor output in a file or database for later analysis, and you need to differentiate the data from different instances of the database manager.

Database Manager Type at Monitored (Server) Node

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	collected	Basic
Resettable	No	
Element Name	server_db2_type	
Element Type	information	
Related Information	• "Configuration NNAME at Monitoring (Server) Node" on page 39	

Description: Identifies the type of database manager being monitored.

Usage: It contains one of the following types of configurations for the database manager:

API Symbolic Constant	**Command Line Processor Output**
sqlf_nt_server	Database Server with local and remote clients
sqlf_nt_stand_req	Database Server with local clients

The API symbolic constants are defined in the include file *sqlutil.h.*

Server Product/Version ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	collected	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database Manager	event_log_header	
Element Name	server_prdid	
Element Type	information	
Related Information	• "Client Product / Version ID" on page 61	

Description: The product and version that is running on the server.

Usage: It is in the form PPPVVRRM, where:

PPP	is SQL
VV	identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version)
RR	identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release)
M	identifies a 1-digit modification level

Server Version

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	collected	Basic
Resettable	No	
Element Name	server_version	
Element Type	information	
Related Information	• "Server Product / Version ID" on page 41	

Description: The version of the server returning the information.

Usage: This field identifies the level of the database server collecting database system monitor information. This allows applications to interpret the data based on the level of the server returning the data. Valid values are:

SQLM_DBMON_VERSION1	Data was returned by DB2 Version 1
SQLM_DBMON_VERSION2	Data was returned by DB2 Version 2
SQLM_DBMON_VERSION5	Data was returned by DB2 Universal Database Version 5

SQLM_DBMON_VERSION5_2	Data was returned by DB2 Universal Database Version 5.2
SQLM_DBMON_VERSION6	Data was returned by DB2 Universal Database Version 6

Service Level

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	service_level	
Element Type	information	
Related Information	• “Product Identification” on page 43	

Description: This is the current corrective service level of the server.

Usage: Used to provide information when requesting service or reporting a problem with DB2 on OS/2. This element will be blank for non-OS/2 systems.

Note: This element is similar to the *corr_serv_lvl* field in the *sqlestat* output. See “Appendix D. DB2 Version 1 sqlestat Users” on page 393 for more information on sqlestat equivalent data elements.

Server Operating System

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	server_platform	
Element Type	information	
Related Information	• “Client Operating Platform” on page 65 • “Database Location” on page 49	

Description: The operating system running the database server.

Usage: This element can be used for problem determination for remote applications. Values for this field can be found in the header file *sqlmon.h*.

Note: This element is similar to the *db_type* field in the *sqlestat* output. See "Appendix D. DB2 Version 1 sqlestat Users" on page 393 for more information on sqlestat equivalent data elements.

Product Name

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	product_name	
Element Type	information	
Related Information	• "Product Identification" on page 43 • "Service Level" on page 42	

Description: Details of the version of the server that is running.

Usage: Used to provide information when requesting service or reporting a problem with DB2 on OS/2. This element will be blank for non-OS/2 systems.

Note: This element is similar to the *product_name* field in the *sqlestat* output. See "Appendix D. DB2 Version 1 sqlestat Users" on page 393 for more information on sqlestat equivalent data elements.

Product Identification

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	component_id	
Element Type	information	
Related Information	• "Product Name" on page 43 • "Service Level" on page 42	

Description: Details of the type of the server that is running.

Usage: Used to provide information when requesting service or reporting a problem with DB2 on OS/2. This element will be blank for non-OS/2 systems.

Note: This element is similar to the *component_id* field in the *sqlestat* output. See "Appendix D. DB2 Version 1 sqlestat Users" on page 393 for more information on sqlestat equivalent data elements.

Status of DB2 Instance

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	db2_status	
Element Type	information	
Related Information	• "Status of Database" on page 48	

Description: The current status of the instance of the database manager.

Usage: You can use this element to determine the state of your database manager instance.

The value returned is always SQLM_DB2_ACTIVE.

Time Zone Displacement

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	collected	Basic
Resettable	No	
Element Name	time_zone_disp	
Element Type	information	
Related Information	• None	

Description: Number of seconds that the local time zone is displaced from Greenwich Mean Time (GMT).

Usage: All time reported by reported by the database system monitor is GMT, this displacement calculates the local time.

Database Identification and Status

The following elements provide identification and status information about the database:

- "Database Name" on page 45
- "Database Path" on page 46
- "Database Activation Timestamp" on page 46
- "Time of Database Connection" on page 47
- "Database Deactivation Timestamp" on page 47

- “Status of Database” on page 48
- “Catalog Node Network Name” on page 48
- “Database Location” on page 49
- “Catalog Node Number” on page 49
- “Last Backup Timestamp” on page 50

Database Name

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl_id_info	Basic
Table Space	tablespace_header	Buffer Pool
	bufferpool	Buffer Pool
Table	table_header	Table
Lock	dbase_lock	Basic
Dynamic SQL	dynsql_list	Basic
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	dbheader_event	
Element Name	db_name	
Element Type	information	
Related Information	• “Resetting Monitor Data” on page 25 • “Last Reset Timestamp” on page 248 • “Input Database Alias” on page 248 • “Database Alias Used by Application” on page 61 • “Database Path” on page 46	

Description: The real name of the database for which information is collected or to which the application is connected. This is the name the database was given when created.

Usage: You may use this element to identify the specific database to which the data applies.

For applications that are not using DB2 Connect to connect to a DRDA host database, you can use this element in conjunction with the *Database Path* monitor element to uniquely identify the database and help relate the different levels of information provided by the monitor.

Database Path

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl_id_info	Basic
Table Space	tablespace_header	Buffer Pool
	bufferpool	Buffer Pool
Table	table_header	Table
Lock	dbase_lock	Basic
Dynamic SQL	dynsql_list	Basic

Resettable	No

Event Type	Logical Data Grouping
Database	dbheader_event

Element Name	db_path
Element Type	information

Related Information

- "Resetting Monitor Data" on page 25
- "Input Database Alias" on page 248
- "Database Name" on page 45

Description: The full path of the location where the database is stored on the monitored system.

Usage: This element can be used with the *Database Name* monitor element to identify the specific database to which the data applies.

Database Activation Timestamp

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Table Space	tablespace_list	Buffer Pool
Table	table_list	Basic

Resettable	No

Element Name	db_conn_time
Element Type	timestamp

Related Information

- "Connection Request Start Timestamp" on page 70
- "Snapshot Time" on page 249
- "Time of Database Connection" on page 47

Description: The date and time of the connection to the database (at the database level, this is the first connection to the database), or when the activate database was issued.

Usage: Use this element with the Database Deactivation Timestamp monitor element to calculate the total connection time.

Time of Database Connection

Event Type	**Logical Data Grouping**
Database	dbheader_event
Connection	connheader_event
Element Name	conn_time
Element Type	timestamp
Related Information	• "Database Activation Timestamp" on page 46 • "Database Deactivation Timestamp" on page 47

Description: The date and time of the connection to the database (at the database level, this is the first connection to the database), or when the activate database was issued.

Usage: Use this element with the Database Deactivation Timestamp monitor element to calculate the elapsed time since:

- The database was active (for information at the database level)
- The connection was active (for information at the connection level).

Database Deactivation Timestamp

Event Type	**Logical Data Grouping**
Database	db_event
Connection	conn_event
Element Name	disconn_time
Element Type	timestamp
Related Information	• None

Description: The date and time that the application disconnected from the database (at the database level, this is the time the last application disconnected).

Usage: Use this element to calculate the elapsed time since:

- The database was active (for information at the database level)
- The connection was active (for information at the connection level).

Status of Database

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Resettable	No	
Element Name **Element Type**	db_status information	
Related Information	• “Status of DB2 Instance” on page 44	

Description: The current status of the database.

Usage: You can use this element to determine the state of your database.

Values for this field are:

API Constant	Description
SQLM_DB_ACTIVE	The database is active.
SQLM_DB_QUIESCE_PEND	The database is in quiesce-pending state. New connections to the database are **not** permitted and new units of work **cannot** be started. Depending on the quiesce request, active units of work will be allowed to complete or will be rolled back immediately.
SQLM_DB_QUIESCED	The database has been quiesced. New connections to the database are **not** permitted and new units of work **cannot** be started.
SQLM_DB_ROLLFWD	A rollforward is in progress on the database.

Catalog Node Network Name

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Resettable	No	
Event Type Database	**Logical Data Grouping** db_event	
Element Name **Element Type**	catalog_node_name information	
Related Information	• None	

Description: The network name of the catalog node. On OS/2, the netbios name of the server where the database is located.

Usage: Use this element to determine the location of a database.

Note: This element is similar to the *node* field in the *sqlestat* output. See "Appendix D. DB2 Version 1 sqlestat Users" on page 393 for more information on sqlestat equivalent data elements.

Database Location

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	db_location	
Element Type	information	
Related Information	• "Server Operating System" on page 42	

Description: The location of the database in relation to the application.

Usage: Determine the relative location of the database server with respect to the application taking the snapshot. Values are:

- SQLM_LOCAL
- SQLM_REMOTE

Note: This element is similar to the *location* field in the *sqlestat* output. See "Appendix D. DB2 Version 1 sqlestat Users" on page 393 for more information on sqlestat equivalent data elements.

Catalog Node Number

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	catalog_node	
Element Type	information	
Related Information	• None	

Description: The node number of the node where the database catalog tables are stored.

Usage: The catalog node is the node where all system catalog tables are stored. All access to system catalog tables must go through this node. See the *Administration Guide* for information on system catalog tables.

Last Backup Timestamp

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	last_backup	
Element Type	timestamp	
Related Information	• None	

Description: The date and time that the latest database backup was completed.

Usage: You may use this element to help you identify a database that has not been backed up recently, or to identify which database backup file is the most recent. If the database has never been backed up, this timestamp is initialized to zero.

Application Identification and Status

The following elements provide information about databases and their related applications.

- "Application Handle (agent ID)" on page 51
- "Application Status" on page 52
- "ID of Code Page Used by Application" on page 55
- "Application Status Change Time" on page 55
- "Application with Oldest Transaction" on page 56
- "Application Name" on page 56
- "Application ID" on page 57
- "Sequence Number" on page 59
- "Authorization ID" on page 60
- "Configuration NNAME of Client" on page 60
- "Client Product/Version ID" on page 61
- "Database Alias Used by Application" on page 61
- "Host Product/Version ID" on page 62
- "Outbound Application ID" on page 62
- "Outbound Sequence Number" on page 63
- "User Login ID" on page 64

- “DRDA Correlation Token” on page 64
- “Client Process ID” on page 65
- “Client Operating Platform” on page 65
- “Client Communication Protocol” on page 66
- “Database Country Code” on page 67
- “Application Agent Priority” on page 67
- “Application Priority Type” on page 68
- “User Authorization Level” on page 68
- “Node Number” on page 69
- “Coordinating Node” on page 70
- “Connection Request Start Timestamp” on page 70
- “Maximum Number of Concurrent Connections” on page 71
- “Connection Request Completion Timestamp” on page 71
- “Previous Unit of Work Completion Timestamp” on page 72
- “Unit of Work Start Timestamp” on page 73
- “Unit of Work Stop Timestamp” on page 74
- “Most Recent Unit of Work Elapsed Time” on page 75
- “Unit of Work Completion Status” on page 75
- “Previous Transaction Stop Time” on page 76
- “Application Idle Time” on page 77

Application Handle (agent ID)

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
Lock	appl_lock	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Statement	stmt_event	
	subsection_event	
Element Name	agent_id	
Element Type	information	
Related Information	• None	

Description: A system-wide **unique** ID for the application. On multi-node systems, where a database is partitioned, this ID will be the same on every node where the application may make a secondary connection.

Usage: The application handle can be used to uniquely identify an active application (application handle is synonymous with agent Id).

Note: The *Application Handle (agent ID)* data element has different behavior depending on your version of DB2. When taking snapshots from DB2 with version SQLM_DBMON_VERSION1 or SQLM_DBMON_VERSION2 to a DB2 Universal Database (Version 5 or greater) database, the *agent_id* returned is not usable as an application identifier, rather it is the *agent_pid* of the agent serving the application. In these cases an *agent_id* is still returned for back-level compatibility, but internally the DB2 Universal Database server will not recognize the value as an *agent_id*.

This value can be used as input to GET SNAPSHOT commands that require an agent Id.

When reading event traces, it can be used to match event records with a given application.

It can also be used as input to the FORCE APPLICATION command or API. On multi-node systems this command can be issued from any node where the application has a connection. Its effect is global

Application Status

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
Lock	appl_lock	Basic
Resettable	No	
Element Name	appl_status	
Element Type	information	
Related Information	• "Application Status Change Time" on page 55 • "Statement Operation" on page 217	

Description: The current status of the application.

Usage: This element can help you diagnose potential application problems. Values for this field are:

API Constant	Description
SQLM_CONNECTPEND	**Database Connect Pending:** The application has initiated a database connection but the request has not yet completed.
SQLM_CONNECTED	**Database Connect Completed:** The application has initiated a database connection and the request has completed.
SQLM_UOWEXEC	**Unit of Work Executing:** The database manager is executing requests on behalf of the unit of work.
SQLM_UOWWAIT	**Unit of Work waiting:** The database manager is waiting on behalf of the unit of work in the application. This status typically means that the system is executing in the application's code.
SQLM_LOCKWAIT	**Lock Wait:** The unit of work is waiting for a lock. After the lock is granted, the status is restored to its previous value.
SQLM_COMMIT_ACT	**Commit Active:** The unit of work is committing its database changes.
SQLM_ROLLBACK_ACT	**Rollback Active:** The unit of work is rolling back its database changes.
SQLM_RECOMP	**Recompiling:** The database manager is recompiling (that is, rebinding) a plan on behalf of the application.
SQLM_COMP	**Compiling:** The database manager is compiling an SQL statement or precompiling a plan on behalf of the application.
SQLM_INTR	**Request Interrupted:** An interrupt of a request is in progress.
SQLM_DISCONNECTPEND	**Database Disconnect Pending:** The application has initiated a database disconnect but the command has not yet completed executing. The application may not have explicitly executed the database disconnect command. The database manager will disconnect from a database if the application ends without disconnecting.
SQLM_TPREP	**Transaction Prepared:** The unit of work is part of a global transaction that has entered the prepared phase of the two-phase commit protocol.
SQLM_THCOMT	**Transaction Heuristically Committed:** The unit of work is part of a global transaction that has been heuristically committed.

API Constant	Description
SQLM_THABRT	**Transaction Heuristically Rolled Back:** The unit of work is part of a global transaction that has been heuristically rolled-back.
SQLM_TEND	**Transaction Ended:** The unit of work is part of a global transaction that has ended but has not yet entered the prepared phase of the two-phase commit protocol.
SQLM_CREATE_DB	**Creating Database:** The agent has initiated a request to create a database and that request has not yet completed.
SQLM_RESTART	**Restarting Database:** The application is restarting a database in order to perform crash recovery.
SQLM_RESTORE	**Restoring Database:** The application is restoring a backup image to the database.
SQLM_BACKUP	**Backing Up Database:** The application is performing a backup of the database.
SQLM_LOAD	**Data Fast Load:** The application is performing a “fast load” of data into the database.
SQLM_UNLOAD	**Data Fast Unload:** The application is performing a “fast unload” of data from the database.
SQLM_IOERROR_WAIT	**Wait to Disable Table space:** The application has detected an I/O error and is attempting to disable a particular table space. The application has to wait for all other active transactions on the table space to complete before it can disable the table space.
SQLM_QUIESCE_TABLESPACE	**Quiescing a Table space:** The application is performing a quiesce table space request.
SQLM_WAITFOR_REMOTE	**Wait for Remote Node:** The application is waiting for a response from a remote node in a partitioned database instance.

ID of Code Page Used by Application

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
Lock	appl_lock	Basic
DCS Application	dcs_appl_info	Basic

Resettable	No

Event Type	**Logical Data Grouping**
Event Log Header	event_log_header
Connection	connheader_event

Element Name	codepage_id
Element Type	information
Related Information	• None

Description: The code page identifier.

Usage: For snapshot monitor data, this is the code page at the node where the monitored application started. This identifier may be used for problem determination for remote applications. You may use this information to ensure that data conversion is supported between the application code page and the database code page (or for DRDA host databases, the host CCSID). For information about supported code pages, see the *Administration Guide.*

For event monitor data, this is the code page of the database for which event data is collected. You can use this element to determine whether your event monitor application is running under a different code page from that used by the database. Data written by the event monitor uses the database code page. If your event monitor application uses a different code page, you may need to perform some character conversion to make the data readable.

Application Status Change Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Unit of Work
Lock	appl_lock	Unit of Work
DCS Application	dcs_appl_info	Unit of Work

Resettable	No
Element Name	status_change_time
Element Type	timestamp
Related Information	• "Resetting Monitor Data" on page 25 • "Application Status" on page 52

Description: The date and time the application entered its current status.

Usage: This element allows you to determine how long an application has been in its current status. If it has been in the same status for a long period of time, this may indicate that it has a problem.

Application with Oldest Transaction

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	appl_id_oldest_xact	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • "Agent ID Holding Lock" on page 181 • "Deadlocks Detected" on page 166	

Description: The application ID (which corresponds to the *agent_id* value from the application snapshot) of the application that has the oldest transaction.

Usage: This element can help you determine which application has the oldest active transaction and is therefore holding the most log space in the database. This application can be forced to free up log space. You should examine the application to determine if it could be modified to commit more frequently.

There are times when there is not a transaction holding up logging, or the oldest transaction does not have an application ID (for example, indoubt transaction or inactive transaction). In these cases, this application's ID is not returned in the data stream.

Application Name

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
Lock	appl_lock	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	appl_name	
Element Type	information	
Related Information	• "Application ID" on page 57 • "ID of Code Page Used by Application" on page 55	

Description: The name of the application running at the client as known to the database manager or DB2 Connect.

Usage: This element may be used with *Application ID* to relate data items with your application.

In a client/server environment, this name is passed from the client to the server to establish the database connection. For DRDA-AS connections, this name is the DRDA external name.

The application name is not available for applications running on the following down-level database client products:

- IBM Extended Services for OS/2

In situations where the client application code page is different from the code page under which the database system monitor is running, you can use *ID of Code Page Used by Application* to help translate *Application Name*.

Application ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
DCS Application	dcs_appl_info	Basic
Lock	appl_lock	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
	connheader_event	
Statement	stmt_event	
Transaction	xaction_event	
Deadlock	dlconn_event	
Element Name	appl_id	
Element Type	information	
Related Information	• “Outbound Application ID” on page 62 • “Client Communication Protocol” on page 66	

Description: This identifier is generated when the application connects to the database at the database manager or when DDCS receives a request to connect to a DRDA database.

Usage: This ID is known on both the client and server, so you can use it to correlate the client and server parts of the application. For DDCS applications, you will also need to use Outbound Application ID to correlate the client and server parts of the application.

This identifier is unique across the network. There are different formats for the application ID, which are dependent on the communication protocol between the client and the server machine on which the database manager and/or DDCS are running. Each of the formats consists of three parts separated by periods.

1. APPC

   Format `Network.LU Name.Application instance`

   Example `CAIBMTOR.OSFDBX0.930131194520`

   Details This application ID is the displayable format of an actual SNA LUWID (Logical Unit-of-Work ID) that flows on the network when an APPC conversation is allocated. APPC-generated application IDs are made up by concatenating the network name, the LU name, and the LUWID instance number, which create a unique label for the client/server application. The network name and LU name can each be a maximum of 8 characters. The application instance corresponds to the 12-decimal-character LUWID instance number.

2. TCP/IP

   Format `*TCPIP.IPAddr.Application instance`

   Example `*TCPIP.A12CF9E8.930131214645`

   Details A TCP/IP-generated application ID is made up by concatenating the string "*TCPIP", the IP address in hexadecimal characters, and a unique identifier for the instance of this application. The IP address is a 32-bit number displayed as a maximum of 8 hexadecimal characters.

3. IPX/SPX

   Format `Netid.nodeid.Application instance`

   Example `C11A8E5C.400011528250.0131214645`

   Details An IPX/SPX-generated application ID is made up by concatenating a character network ID (8 hexadecimal characters), a node id (12 hexadecimal characters), and a unique identifier for the instance of the application. The application instance corresponds to a 10-decimal-character time stamp of the form `MMDDHHMMSS`.

4. NetBIOS

   Format `*NETBIOS.nname.Application instance`

   Example `*NETBIOS.SBOIVIN.930131214645`

Details	A NetBIOS application ID is made up by concatenating the string "*NETBIOS", the nname defined in the client's database configuration file, and a unique identifier for the instance of this application.

5. Local Applications

Format	`*LOCAL.DB2 instance.Application instance`
Example	`*LOCAL.DB2INST1.930131235945`
Details	The application ID generated for a local application is made up by concatenating the string `*LOCAL`, the name of the DB2 instance, and a unique identifier for the instance of this application.

Use *Client Communication Protocol* to determine which communications protocol the connection is using and, as a result, the format of the *application ID*.

Sequence Number

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
	connheader_event	
Statement	stmt_event	
Transaction	xaction_event	
Deadlock	dlconn_event	
Element Name	sequence_no	
Element Type	information	
Related Information	• None	

Description: This element is reserved for future use. In this release, its value always be "0001". It may contain different values in future releases of the product.

Authorization ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
Lock	appl_lock	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	auth_id	
Element Type	information	
Related Information	• "Application Name" on page 56	

Description: The authorization ID of the user who invoked the application that is being monitored. On a DB2 Connect gateway node, this is the user's authorization ID on the host.

Usage: You can use this element to determine who invoked the application.

Configuration NNAME of Client

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	client_nname	
Element Type	information	
Related Information	• "Configuration NNAME at Monitoring (Server) Node" on page 39	

Description: The *nname* in the database manager configuration file at the client node.

Usage: You can use this element to identify the client node that is running the application.

Client Product/Version ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	client_prdid	
Element Type	information	
Related Information	• "Server Product / Version ID" on page 41	

Description: The product and version that is running on the client.

Usage: You can use this element to identify the product and code version of the database client. It is in the form PPPVVRRM, where:

- PPP identifies the product, which is "SQL" for the DB2 products
- VV identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version)
- RR identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release)
- M identifies a 1-digit modification level.

Database Alias Used by Application

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_id_info	Basic
Lock	appl_lock	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	client_db_alias	
Element Type	information	
Related Information	• All other database-level information • All other application-level information • "Last Reset Timestamp" on page 248 • "Input Database Alias" on page 248 • "Database Name" on page 45	

Description: The alias of the database provided by the application to connect to the database.

Usage: This element can be used to identify the actual database that the application is accessing. The mapping between this name and *Database Name* could be done by using the database directories at the client node and the database manager server node.

This is the alias defined within the database manager where the database connection request originated.

This element can also be used to help you determine the authentication type, since different database aliases can have different authentication types.

Host Product/Version ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	host_prdid	
Element Type	information	
Related Information	• None	

Description: The product and version that is running on the server.

Usage: Used to identify the product and code version of the DRDA host database product. It is in the form PPPVVRRM, where:

- PPP identifies the host DRDA product
  - ARI for DB2 for VSE & VM
  - DSN for DB2 for MVS/ESA
  - QSQ for DB2 Universal Database for AS/400
  - SQL for other DB2 products.
- VV identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version)
- RR identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release)
- M identifies a 1-digit modification level

Outbound Application ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	outbound_appl_id	
Element Type	information	
Related Information	• "Application ID" on page 57	

Description: This identifier is generated when the application connects to the DRDA host database. It is used to connect the DB2 Connect gateway to the host, while the *Application ID* is used to connect a client to the DB2 Connect gateway.

Usage: You may use this element in conjunction with *Application ID* to correlate the client and server parts of the application information.

This identifier is unique across the network.

Format	`Network.LU Name.Application instance`
Example	`CAIBMTOR.OSFDBM0.930131194520`
Details	This application ID is the displayable format of an actual SNA LUWID (Logical Unit-of-Work ID) that flows on the network when an APPC conversation is allocated. APPC-generated application IDs are made up by concatenating the network name, the LU name, and the LUWID instance number, which creates a unique label for the client/server application. The network name and LU name can each be a maximum of 8 characters. The application instance corresponds to the 12-decimal-character LUWID instance number.

Outbound Sequence Number

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name **Element Type**	outbound_sequence_no information	
Related Information	• None	

Description: This element is reserved for future use. In this release, its value will always be “0001”. It may contain different values in future releases of the product.

User Login ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_info	Basic
	appl	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	execution_id	
Element Type	information	
Related Information	• "Authorization ID" on page 60	

Description: The ID that the user specified when logging in to the operating system. This ID is distinct from Authorization ID, which the user specifies when connecting to the database.

Usage: You can use this element to determine the operating system userid of the individual running the application that you are monitoring.

DRDA Correlation Token

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_info	Basic
	appl	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	corr_token	
Element Type	information	
Related Information	• None	

Description: The DRDA AS correlation token.

Usage: The DRDA correlation token is used for correlating the processing between the application server and the application requester. It is an identifier dumped into logs when errors arise, that you can use to identify the conversation that is in error. In some cases, it will be the LUWID of the conversation.

If communications are not using DRDA, this element is blank.

If you are using the database system monitor APIs, note that the API constant SQLM_APPLID_SZ is used to define the length of this element.

Client Process ID

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
	appl	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	client_pid	
Element Type	information	
Related Information	• None	

Description: The process ID of the client application that made the connection to the database.

Usage: You can use this element to correlate monitor information such as CPU and I/O time to your client application.

In the case of a DRDA AS connection, this element will be set to 0.

Client Operating Platform

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
	appl	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	client_platform	
Element Type	information	
Related Information	• “Server Operating System” on page 42	

Description: The operating system on which the client application is running.

Usage: This element can be used for problem determination for remote applications. Values for this field can be found in the header file *sqlmon.h*.

Client Communication Protocol

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
	appl	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Element Name	client_protocol	
Element Type	information	
Related Information	• None	

Description: The communication protocol that the client application is using to communicate with the server.

Usage: This element can be used for problem determination for remote applications. Values for this field are:

API Constant	Communication Protocol
SQLM_PROT_UNKNOWN	(note 1)
SQLM_PROT_LOCAL	none (note 2)
SQLM_PROT_APPC	APPC
SQLM_PROT_TCPIP	TCP/IP
SQLM_PROT_IPXSPX	IPX/SPX
SQLM_PROT_NETBIOS	NETBIOS

Notes:

1. The client is communicating using an unknown protocol. This value will only be returned if future clients connect with a down-level server.
2. The client is running on the same node as the server and no communications protocol is in use.

Database Country Code

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
	appl	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Event Log Header	event_log_header	
Connection	connheader_event	
Element Name	country_code	
Element Type	information	
Related Information	• None	

Description: The country code of the database for which the monitor data is collected.

Usage: Country code information is recorded in the database configuration file (see the *Administration Guide*).

For DRDA AS connections, this element will be set to 0.

Application Agent Priority

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Element Name	appl_priority	
Element Type	information	
Related Information	• "Application Priority Type" on page 68	

Description: The priority of the agents working for this application.

Usage: You can use this element to check if applications are running with the expected priorities. Application priorities can be set by an administrator. They can be changed by the governor utility (**db2gov**).

The governor is used by DB2 to monitor and change the behavior of applications running against a database. This information is used to schedule applications and balance system resources.

A governor daemon collects statistics about the applications by taking snapshots. It checks these statistics against the rules governing applications running on that database. If the governor detects a rule violation, it takes the appropriate action. These rules and actions were specified by you in the governor configuration file.

If the action associated with a rule is to change an application's priority, the governor changes the priority of the agents in the partition where the violation was detected.

See the *Administration Guide* for more information on the governor.

Application Priority Type

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Element Name	appl_priority_type	
Element Type	information	
Related Information	• "Query Cost Estimate" on page 227 • "Application Agent Priority" on page 67	

Description: Operating system priority type for the agent working on behalf of the application.

Usage: Dynamic priority is recalculated by the operating system based on usage. Static priority does not change.

User Authorization Level

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
	appl_info	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Element Name	authority_lvl	
Element Type	information	
Related Information	• None	

Description: The highest authority level granted to an application.

Usage: The operations allowed by an application are granted either directly or indirectly in the *sql.h*.

These are the authorizations granted explicitly to a user:

- SQL_SYSADMIN
- SQL_DBADM
- SQL_CREATETAB
- SQL_BINDADD
- SQL_CONNECT
- SQL_CREATE_NOT_FENC
- SQL_SYSCTRL
- SQL_SYSMAINT

The following are indirect authorizations inherited from group or public:

- SQL_SYSADM_GRP
- SQL_DBADM_GRP
- SQL_CREATETAB_GRP
- SQL_BINDADD_GRP
- SQL_CONNECT_GRP
- SQL_CREATE_NOT_FENC_GRP
- SQL_SYSCTRL_GRP
- SQL_SYSMAINT_GRP

See the *Administration Guide* for detailed information on authority levels.

Node Number

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	collected	Basic
	fcm_node	Basic
Table Space	rollforward	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	connheader_event	
Overflow	overflow_event	
Element Name	node_number	
Element Type	information	
Related Information	• None	

Description: The number assigned to the node in the *db2nodes.cfg* file.

Usage: This value identifies the current node number, which can be used when monitoring multiple nodes.

Coordinating Node

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Element Name	coord_node	
Element Type	information	
Related Information	• None	

Description: In a multi-node system, the node number of the node where the application connected or attached to the instance.

Usage: Each connected application is served by one coordinator node.

Connection Request Start Timestamp

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Element Name	appl_con_time	
Element Type	timestamp	
Related Information	• “Connection Request Completion Timestamp” on page 71 • All information related to the application	

Description: The date and time that an application started a connection request.

Usage: Use this element to determine when the application started its connection request to the database.

Maximum Number of Concurrent Connections

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	connections_top	
Element Type	water mark	
Related Information	• "Remote Connections To Database Manager" on page 80 • "Local Connections" on page 82	

Description: The highest number of simultaneous connections to the database since the database was activated.

Usage: You may use this element to evaluate the setting of the *maxappls* configuration parameter, which is described in the *Administration Guide*.

If the value of this element is the same as the *maxappls* parameter, it is likely that some database connection requests were rejected, since *maxappls* limits the number of database connections allowed.

The current number of connections at the time the snapshot was taken can be calculated using the following formula:

```
remote connections to database manager + local connections
```

Connection Request Completion Timestamp

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Element Name	conn_complete_time	
Element Type	timestamp	
Related Information	• "Connection Request Start Timestamp" on page 70 • All information related to the application	

Description: The date and time that a connection request was granted.

Usage: Use this element to determine when a connection request to the database was granted.

Previous Unit of Work Completion Timestamp

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Unit of Work
DCS Application	dcs_appl	Unit of Work
Resettable	No	
Element Name	prev_uow_stop_time	
Element Type	timestamp	
Related Information	• “Resetting Monitor Data” on page 25 • “Unit of Work Start Timestamp” on page 73 • “Unit of Work Stop Timestamp” on page 74 • “Connection Request Completion Timestamp” on page 71	

Description: This is the time the unit of work completed.

Usage: You may use this element with *Unit of Work Stop Timestamp* to calculate the total elapsed time between COMMIT/ROLLBACK points, and with *Unit of Work Start Timestamp* to calculate the time spent in the application between units of work. The time of one of the following:

- For applications currently within a unit of work, this is the time that the latest unit of work completed.
- For applications not currently within a unit of work (the application has completed a unit of work, but not yet started a new one), this is the stop time of the last unit of work that completed prior to the one that just completed. The stop time of the one just completed is indicated “Unit of Work Stop Timestamp” on page 74.
- For applications within their first unit of work, this is the database connection request completion time.

Unit of Work Start Timestamp

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Unit of Work
DCS Application	dcs_appl	Unit of Work
Resettable	No	
Element Name	uow_start_time	
Element Type	timestamp	
Related Information	• "Resetting Monitor Data" on page 25 • "Unit of Work Stop Timestamp" on page 74 • "Previous Unit of Work Completion Timestamp" on page 72 • "Connection Request Completion Timestamp" on page 71	

Description: The date and time that the unit of work first required database resources.

Usage: This resource requirement occurs at the first SQL statement execution of that unit of work:

- For the first unit of work, it is the time of the first database request (SQL statement execution) after *Connection Request Completion Timestamp.*
- For subsequent units of work, it is the time of the first database request (SQL statement execution) after the previous COMMIT or ROLLBACK.

Note: The *SQL Reference* defines the boundaries of a unit of work as the COMMIT or ROLLBACK points.

The database system monitor excludes the time spent between the COMMIT/ROLLBACK and the next SQL statement from its definition of a unit of work. This measurement method reflects the time spent by the database manager in processing database requests, separate from time spent in application logic before the first SQL statement of that unit of work. The unit of work elapsed time does include the time spent running application logic between SQL statements within the unit of work.

You may use this element with *Unit of Work Stop Timestamp* to calculate the total elapsed time of the unit of work and with *Previous Unit of Work Completion Timestamp* to calculate the time spent in the application between units of work.

You can use the *Unit of Work Stop Timestamp* and the *Previous Unit of Work Completion Timestamp* to calculate the elapsed time for the SQL Reference's definition of a unit of work.

Unit of Work Stop Timestamp

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Unit of Work
DCS Application	dcs_appl	Unit of Work
Resettable	No	
Element Name	uow_stop_time	
Element Type	timestamp	
Related Information	• "Resetting Monitor Data" on page 25 • "Unit of Work Start Timestamp" on page 73 • "Previous Unit of Work Completion Timestamp" on page 72 • "Connection Request Completion Timestamp" on page 71	

Description: The date and time that the most recent unit of work completed, which occurs when database changes are committed or rolled back.

Usage: You may use this element with *Previous Unit of Work Completion Timestamp* to calculate the total elapsed time between COMMIT/ROLLBACK points, and with *Unit of Work Start Timestamp* to calculate the elapsed time of the latest unit of work.

The timestamp contents will be set as follows:

- When the application has completed a unit of work and has not yet started a new one (as defined in *Unit of Work Start Timestamp).* this element will be a valid, non-zero timestamp
- When the application is currently executing a unit of work, this element will contain zeros
- When the application first connects to the database, this element is set to *Connection Request Completion Timestamp.*

As a new unit of work is started, the contents of this element are moved to *Previous Unit of Work Completion Timestamp.*

Most Recent Unit of Work Elapsed Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Unit of Work	appl	Unit of Work
DCS Unit of Work	dcs_appl	Unit of Work
Resettable	No	
Element Name	uow_elapsed_time	
Element Type	time	
Related Information	• "Communication Errors" on page 266 • "Communication Error Time" on page 267	

Description: The elapsed execution time of the most recently completed unit of work.

Usage: Use this element as an indicator of the time it takes for units of work to complete.

Unit of Work Completion Status

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Unit of Work
DCS Application	dcs_appl	Unit of Work
Resettable	No	

Event Type	**Logical Data Grouping**
Transaction	xaction_event
Element Name	uow_comp_status
Element Type	information
Related Information	• "Resetting Monitor Data" on page 25

Description: The status of the unit of work and how it stopped.

Usage: You may use this element to determine if the unit of work ended due to a deadlock or abnormal termination. It may have been:

- Committed due to a commit statement
- Rolled back due to a rollback statement
- Rolled back due to a deadlock
- Rolled back due to an abnormal termination
- Committed at normal application termination.
- Unknown as a result of a FLUSH EVENT MONITOR command for which units of work were in progress.

Note: API users should refer to the header file (*sqlmon.h*) containing definitions of database system monitor constants.

Unit of Work Status

Event Type	**Logical Data Grouping**
Transaction	xaction_event
Element Name	uow_status
Element Type	information
Related Information	• “Resetting Monitor Data” on page 25

Description: The status of the unit of work.

Usage: You may use this element to determine the status of a unit of work.

Previous Transaction Stop Time

Event Type	**Logical Data Grouping**
Transaction	xaction_event
Element Name	prev_stop_time
Element Type	timestamp
Related Information	• None

Description: The completion time of the last unit of work.

Usage: You may use this element to calculate the time spent in the application between units of work.

This is the unit of work that completed prior to the unit of work for which this transaction event is generated.

For applications within their first unit of work, this is the database connection request completion time.

Application Idle Time

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement
DCS Application/lines>	dcs_appl	Statement
Resettable	No	
Element Name	appl_idle_time	
Element Type	information	
Related Information	• "Query Cost Estimate" on page 227 • "Application Agent Priority" on page 67 • "Application Priority Type" on page 68	

Description: Number of seconds since an application has issued any requests to the server. This includes applications that have not terminated a transaction, for example not issued a commit or rollback.

Usage: This information can be used to implement applications that force users that have been idle for a specified number of seconds.

DB2 Agent Information

The following database system monitor elements provide information about agents:

- "Process or Thread ID"
- "Coordinator Agent" on page 78

Process or Thread ID

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	agent	Statement
Resettable	No	
Element Name	agent_pid	
Element Type	information	
Related Information	• "Coordinator Agent" on page 78	

Description: The process Id (UNIX systems) or thread Id (OS/2 or Windows systems) of a DB2 agent.

Usage: You can use this element to link database system monitor information to other sources of diagnostic information, such as system traces. You can also use it to monitor how agents working for a database application use system resources.

Coordinator Agent

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_info	Basic
Resettable	No	
Element Name	coord_agent_pid	
Element Type	information	
Related Information	• “Process or Thread ID” on page 77	

Description: The process Id (UNIX systems) or thread Id (OS/2 or Windows systems) of the coordinator agent for the application.

Usage: You can use this element to link database system monitor information to other sources of diagnostic information, such as system traces.

Database Manager Configuration

The following elements provide database manager configuration information.

Agents and Connections

An agent is a process or thread that carries out the requests made by a client application. Each connected application is served by exactly 1 **coordinator agent** and possibly, a set of subordinator agents or **subagents**. Subagents are used for parallel SQL processing in partitioned databases and on SMP machines. Agents are classified as follows:

- **Coordinator agent**

  This is the initial agent to which a local or remote application connects. There is one coordinator agent dedicated to each database connection or instance attachment. The maximum number of coordinating agents per partition is controlled by the *max_coordagents* configuration parameter.
- **Subagent**

  In partitioned databases, additional agents can be enlisted by the coordinator agent to speed up SQL processing. Subagents are selected from the agent pool and are returned there when their work is done. The size of the agent pool is controlled by the *num_poolagents* configuration parameter.
- **Associated agent**

  A coordinator or subagent that is doing work for an application is associated with that application. After it is finished an application’s work, it goes into the agent pool as an associated agent. If the application attempts to do more work, DB2 will search the agent pool for an agent already associated with the application and assign the work to it. If none is found, DB2 will attempt to get an agent to satisfy the request by:

1. Choosing an idle agent that is not associated with an application.
2. Creating an agent, if an idle agent is not available.
3. Finding an agent that is associated with another application. For example, if an agent cannot be created because *maxagents* has been reached, DB2 will try to take an idle agent associated with another application. This is referred to as a **stolen agent**.

- **Primed agent**

  A gateway agent in the DRDA connections pool that is connected to a DRDA database in anticipation of work on the remote database.

The *maxagents* configuration parameter defines the maximum number of agents, regardless of type, that can exist for an instance. The *maxagents* value does not create any agents. The initial number of agents that are created in the agent pool at DB2START is determined by the *num_initagents* configuration parameter.

Assuming no idle agents, each connection creates a new agent, unless *max_coordagents* has been reached. If subagents are not used, *max_coordagents* equals *maxagents*. If subagents are used, some combination of coordinator agents and subagents could reach *maxagents*.

When an agent is assigned work, it attempts to obtain a token or permission to process the transaction. The database manager controls the number of tokens available using the *maxcagents* configuration parameter. If a token is not available, the agent will sleep until one becomes available, at which time the requested work will be processed. This allows you to use *maxcagents* to control the load, or number of concurrently executing transactions, the server handles.

The following elements provide agent and connection information:

- “Remote Connections To Database Manager” on page 80
- “Remote Connections Executing in the Database Manager” on page 81
- “Local Connections” on page 82
- “Local Connections Executing in the Database Manager” on page 82
- “Local Databases with Current Connects” on page 83
- “Connects Since Database Activation” on page 84
- “Applications Connected Currently” on page 85
- “Applications Executing in the Database Currently” on page 85
- “Agents Registered” on page 86
- “Agents Waiting for a Token” on page 86

- “Maximum Number of Agents Registered” on page 87
- “Maximum Number of Agents Waiting” on page 87
- “Number of Idle Agents” on page 88
- “Agents Assigned From Pool” on page 88
- “Agents Created Due to Empty Agent Pool” on page 89
- “Maximum Number of Coordinating Agents” on page 90
- “Stolen Agents” on page 90
- “Maximum Number of Associated Agents” on page 91
- “Committed Private Memory” on page 91
- “Secondary Connections” on page 92
- “Number of Associated Agents” on page 92
- “Maximum Agent Overflows” on page 93
- “Total Inactive DRDA Agents” on page 93
- “Connection Switches” on page 94

Remote Connections To Database Manager

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	rem_cons_in	
Element Type	gauge	
Related Information	• “Remote Connections Executing in the Database Manager” on page 81 • “Local Connections” on page 82 • “Local Connections Executing in the Database Manager” on page 82	

Description: The current number of connections initiated from remote clients to the instance of the database manager that is being monitored.

Usage: Shows the number of connections from remote clients to databases in this instance. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage. This number does not include applications that were initiated from the same instance as the database manager.

When used in conjunction with the Local Connections monitor element, these elements can help you adjust the setting of the *max_coordagents* configuration parameter, described in the *Administration Guide*.

Remote Connections Executing in the Database Manager

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	rem_cons_in_exec	
Element Type	gauge	
Related Information	• "Remote Connections To Database Manager" on page 80 • "Local Connections" on page 82 • "Local Connections Executing in the Database Manager" on page 82	

Description: The number of remote applications that are currently connected to a database and are currently processing a unit of work within the database manager instance being monitored.

Usage: This number can help you determine the level of concurrent processing occurring on the database manager. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage. This number does not include applications that were initiated from the same instance as the database manager.

When used in conjunction with the Local Connections Executing in the Database Manager monitor element, this element can help you adjust the setting of the *maxcagents* configuration parameter, described in the *Administration Guide*.

Local Connections

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	local_cons	
Element Type	gauge	
Related Information	• "Remote Connections To Database Manager" on page 80 • "Remote Connections Executing in the Database Manager" on page 81 • "Local Connections Executing in the Database Manager" on page 82	

Description: The number of local applications that are currently connected to a database within the database manager instance being monitored.

Usage: This number can help you determine the level of concurrent processing occurring in the database manager. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage.

This number only includes applications that were initiated from the same instance as the database manager. The applications are connected, but may or may not be executing a unit of work in the database.

When used in conjunction with the Remote Connections To Database Manager monitor element, this element can help you adjust the setting of the *maxagents* configuration parameter, described in the *Administration Guide*.

Local Connections Executing in the Database Manager

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	local_cons_in_exec	
Element Type	gauge	
Related Information	• "Remote Connections To Database Manager" on page 80 • "Remote Connections Executing in the Database Manager" on page 81 • "Local Connections" on page 82	

Description: The number of local applications that are currently connected to a database within the database manager instance being monitored and are currently processing a unit of work.

Usage: This number can help you determine the level of concurrent processing occurring in the database manager. This value will change frequently, so you may need to sample it at specific intervals over an extended period of time to get a realistic view of system usage. This number only includes applications that were initiated from the same instance as the database manager.

When used in conjunction with the Remote Connections Executing in the Database Manager monitor element, this element can help you adjust the setting of the *maxcagents* configuration parameter, described in the *Administration Guide*.

Local Databases with Current Connects

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	con_local_dbases	
Element Type	gauge	
Related Information	• None	

Description: The number of local databases that have applications connected.

Usage: This value gives an indication of how many database information records you can expect when gathering data at the database level.

The applications can be running locally or remotely, and may or may not be executing a unit of work within the database manager

Connects Since Database Activation

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	total_cons	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Database Activation Timestamp" on page 46 • "Applications Connected Currently" on page 85 • "Applications Executing in the Database Currently" on page 85 • "Secondary Connections" on page 92	

Description: Indicates the number of connections to the database since the first connect, activate, or last reset (coordinator agents).

Usage: You can use this element in conjunction with the Database Activation Timestamp and the Start Database Manager Timestamp monitor elements to calculate the frequency at which applications have connected to the database.

If the frequency of connects is low, you may want to explicitly activate the database using the ACTIVATE DATABASE command before connecting any other application, because of the extra overhead that is associated with the first connect to a database (for example, initial buffer pool allocation). This will result in subsequent connects being processed at a higher rate.

Note: When you reset this element, its value is set to the number of applications that are currently connected, not to zero.

Applications Connected Currently

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Lock	dbase_lock	Basic
Resettable	No	
Element Name	appls_cur_cons	
Element Type	gauge	
Related Information	• "Applications Executing in the Database Currently" on page 85 • "Connects Since Database Activation" on page 84 • "Remote Connections To Database Manager" on page 80 • "Local Connections" on page 82	

Description: Indicates the number of applications that are currently connected to the database.

Usage: You may use this element to help you understand the level of activity within a database and the amount of system resource being used.

It can help you adjust the setting of the *maxappls* and *max_coordagents* configuration parameters, which are described in the *Administration Guide*. For example, its value is always the same as *maxappls*, you may want to increase the value of *maxappls*. See the *Remote Connections To Database Manager* and the *Local Connections* monitor elements for more information.

Applications Executing in the Database Currently

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	appls_in_db2	
Element Type	gauge	
Related Information	• "Applications Connected Currently" on page 85 • "Connects Since Database Activation" on page 84 • "Remote Connections Executing in the Database Manager" on page 81 • "Local Connections Executing in the Database Manager" on page 82 • "Current Agents Waiting On Locks" on page 179	

Description: Indicates the number of applications that are currently connected to the database, and for which the database manager is currently processing a request.

Usage: You can use this element to understand how many of the database manager agent tokens are being used by applications connected to this database. If the sum of *Remote Connections Executing in the Database Manager* and *Local Connections Executing in the Database Manager* is equal to the value of the *maxcagents* configuration parameter, you may want to increase the value of that parameter, as described in the *Administration Guide*.

Agents Registered

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	agents_registered	
Element Type	gauge	
Related Information	• "Maximum Number of Agents Registered" on page 87	

Description: The number of agents registered in the database manager instance that is being monitored (coordinator agents and subagents).

Usage: You can use this element to help evaluate your setting for the *maxagents* configuration parameter.

Agents Waiting for a Token

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	agents_waiting_on_token	
Element Type	gauge	
Related Information	• "Agents Registered" on page 86	

Description: The number of agents waiting for a token so they can execute a transaction in the database manager.

Usage: You can use this element to help evaluate your setting for the *maxcagents* configuration parameter.

Each application has a dedicated coordinator agent to process database requests within the database manager. Each agent has to get a token before it can execute a transaction. The maximum number of agents that can execute database manager transactions is limited by the configuration parameter *maxcagents*. For more information about this parameter, see the *Administration Guide*.

Maximum Number of Agents Registered

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	agents_registered_top	
Element Type	water mark	
Related Information	• "Agents Registered" on page 86 • "Maximum Number of Agents Waiting" on page 87	

Description: The maximum number of agents that the database manager has ever registered, at the same time, since it was started (coordinator agents and subagents).

Usage: You may use this element to help you evaluate your setting of the *maxagents* configuration parameter, described in the *Administration Guide*.

The number of agents registered at the time the snapshot was taken is recorded by Agents Registered.

Maximum Number of Agents Waiting

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	agents_waiting_top	
Element Type	water mark	
Related Information	• "Agents Waiting for a Token" on page 86 • "Maximum Number of Agents Registered" on page 87	

Description: The maximum number of agents that have ever been waiting for a token, at the same time, since the database manager was started.

Usage: You may use this element to help you evaluate your setting of the *maxcagents* configuration parameter, described in the *Administration Guide*.

The number of agents waiting for a token at the time the snapshot was taken is recorded by *Agents Waiting for a Token*.

If the *maxcagents* parameter is set to its default value (-1), no agents should wait for a token and the value of this monitor element should be zero.

Number of Idle Agents

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	idle_agents	
Element Type	gauge	
Related Information	• "Maximum Number of Agents Registered" on page 87 • "Maximum Number of Agents Waiting" on page 87 • "Agents Registered" on page 86	

Description: The number of agents in the agent pool that are currently unassigned to an application and are, therefore, "idle".

Usage: You can use this element to help set the *num_poolagents* configuration parameter. Having idle agents available to service requests for agents can improve performance. See the *Administration Guide* for more information.

Agents Assigned From Pool

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	agents_from_pool	
Element Type	counter	
Related Information	• "Agents Created Due to Empty Agent Pool" on page 89 • "Maximum Number of Coordinating Agents" on page 90	

Description: The number of agents assigned from the agent pool.

Usage: This element can be used with *Agents Created Due to Empty Agent Pool* to determine how often an agent must be created because the pool is empty.

If the ratio of

```
Agents Created Due to Empty Agent Pool / Agents Assigned From Pool
```

is high, it may indicate that the *num_poolagents* configuration parameter should be increased. A low ratio suggests that *num_poolagents* is set too high, and that some of the agents in the pool are rarely used and are wasting system resources.

A high ratio can indicate that the overall workload for this node is too high. You can adjust the workload by lowering the maximum number of coordinating agents specified by the *maxcagents* configuration parameter, or by redistributing data among the nodes.

See the *Administration Guide* for more information on the Agent Pool Size (*num_poolagents*) and Maximum Number of Concurrent Coordinating Agents (*maxcagents*) configuration parameters.

Agents Created Due to Empty Agent Pool

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	agents_created_empty_pool	
Element Type	counter	
Related Information	• "Agents Assigned From Pool" on page 88 • "Maximum Number of Coordinating Agents" on page 90	

Description: The number of agents created because the agent pool was empty.

Usage: In conjunction with Agents Assigned From Pool, you can calculate the ratio of

```
Agents Created Due to Empty Agent Pool / Agents Assigned From Pool
```

See "Agents Assigned From Pool" on page 88 for information on using this element.

Maximum Number of Coordinating Agents

Snapshot Level	**Logical Data Grouping**	Basic
Database Manager	db2	Basic
Database	dbase	
Resettable	No	
Element Name	coord_agents_top	
Element Type	water mark	
Related Information	• "Agents Assigned From Pool" on page 88 • "Agents Created Due to Empty Agent Pool" on page 89	

Description: The maximum number of coordinating agents working at one time.

Usage: If the peak number of coordinating agents represents too high a workload for this node, you can reduce the number that can be concurrently executing a transaction by changing the *maxcagents* configuration parameter.

See the *Administration Guide* for more information on the Maximum Number of Concurrent Coordinating Agents (*maxcagents*) configuration parameter.

Stolen Agents

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Application	appl	Basic
Resettable	Yes	
Element Name	agents_stolen	
Element Type	counter	
Related Information	• "Number of Agents Working on a Statement" on page 237	

Description: The number of times that agents are stolen from an application. Agents are stolen when an idle agent associated with an application is reassigned to work on a different application.

Usage: This element can be used in conjunction with *"Maximum Number of Associated Agents" on page 91* to evaluate the load that this application places on the system.

Maximum Number of Associated Agents

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic
Resettable	No	
Element Name	associated_agents_top	
Element Type	water mark	
Related Information	• "Agents Assigned From Pool" on page 88 • "Agents Created Due to Empty Agent Pool" on page 89	

Description: The maximum number of subagents associated with this application.

Usage: If the peak number of subagents is close to the *num_poolagents* configuration parameter, this might indicate too high a workload for this node.

See the *Administration Guide* for more information on the Agent Pool Size (*num_poolagents*) configuration parameter.

Committed Private Memory

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Resettable	No	
Element Name	comm_private_mem	
Element Type	gauge	
Related Information	• none	

Description: The amount of private memory that the instance of the database manager has currently committed at the time of the snapshot.

Usage: You can use this element to help set the *min_priv_mem* configuration parameter (see the *Administration Guide*) to ensure you have enough private memory available. This element is returned for all platforms, but tuning can only be accomplished on platforms where DB2 uses threads (such as OS/2 and Windows NT).

Secondary Connections

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Resettable	No	
Element Name	total_sec_cons	
Element Type	counter	
Related Information	• "Connects Since Database Activation" on page 84 • "Start Database Manager Timestamp" on page 39 • "Database Activation Timestamp" on page 46	

Description: The number of connections made by a subagent to the database at the node.

Usage: You can use this element in conjunction with the Connects Since Database Activation, Database Activation Timestamp, and the Start Database Manager Timestamp monitor elements to calculate the frequency at which applications have connected to the database.

Number of Associated Agents

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl_info	Basic
Resettable	No	
Element Name	num_assoc_agents	
Element Type	gauge	
Related Information	• "Agents Assigned From Pool" on page 88 • "Agents Created Due to Empty Agent Pool" on page 89 • "Maximum Number of Associated Agents" on page 91 • "Maximum Agent Overflows" on page 93	

Description: At the application level, this is the number of subagents associated with an application. At the database level, it is the number of subagents for all applications.

Usage: You can use this element to help evaluate your settings for your agent configuration parameters.

Maximum Agent Overflows

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Resettable	No	
Element Name	max_agent_overflows	
Element Type	gauge	
Related Information	• "Agents Assigned From Pool" on page 88 • "Agents Created Due to Empty Agent Pool" on page 89 • "Maximum Number of Associated Agents" on page 91 • "Number of Associated Agents" on page 92	

Description: The number of times a request to create a new agent was received when the *maxagents* configuration parameter had already been reached.

Usage: If agent creation requests are still being received when the *maxagents* configuration parameter has been reached, this might indicate too high a workload for this node.

See the *Administration Guide* for more information on the Maximum Number of Agents (*maxagents*) configuration parameter.

Total Inactive DRDA Agents

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Resettable	No	
Element Name	inactive_gw_agents	
Element Type	gauge	
Related Information	• "Maximum Number of Agents Registered" on page 87 • "Agents Registered" on page 86 • "Number of Associated Agents" on page 92 • "Maximum Agent Overflows" on page 93 • "Connection Switches" on page 94	

Description: The number of DRDA agents in the DRDA connections pool that are primed with a connection to a DRDA database, but are inactive.

Usage: Using this element over time will help determine if the number of agents allocated to the connections pool is adequate.

Connection Switches

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Element Name	num_gw_conn_switches	
Element Type	gauge	
Related Information	• "Maximum Number of Agents Registered" on page 87 • "Agents Registered" on page 86 • "Number of Associated Agents" on page 92 • "Maximum Agent Overflows" on page 93 • "Secondary Connections" on page 92 • "Total Inactive DRDA Agents" on page 93	

Description: The number of times that an agent from the agents pool was primed with a connection and was stolen for use with a different DRDA database.

Usage: Use this element in conjunction with "Total Inactive DRDA Agents" on page 93 to determine if the size of the agent pool should be increased.

Sort

The following elements provide information about the database manager sort work performed:

- "Total Sort Heap Allocated" on page 95
- "Post Threshold Sorts" on page 96
- "Piped Sorts Requested" on page 97
- "Piped Sorts Accepted" on page 97
- "Total Sorts" on page 98
- "Total Sort Time" on page 99
- "Sort Overflows" on page 100
- "Active Sorts" on page 101

Total Sort Heap Allocated

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Database	dbase	Basic
Resettable	No	
Element Name	sort_heap_allocated	
Element Type	gauge	
Related Information	• "Total Sorts" on page 98	

Description: The total number of allocated pages of sort heap space for all sorts at the level chosen and at the time the snapshot was taken.

Usage: The amount of memory allocated for each sort may be some or all of the available sort heap size. Sort heap size is the amount of memory available for each sort as defined in the database configuration parameter *sortheap*.

It is possible for a single application to have concurrent sorts active. For example, in some cases a SELECT statement with a subquery can cause concurrent sorts.

Information may be collected at two levels:

- At the database manager level, it represents the sum of sort heap space allocated for all sorts in all active databases in the database manager
- At the database level, it represents the sum of the sort heap space allocated for all sorts in a database.

Normal memory estimates do not include sort heap space. If excessive sorting is occurring, the extra memory used for the sort heap should be added to the base memory requirements for running the database manager. Generally, the larger the sort heap, the more efficient the sort. Appropriate use of indexes can reduce the amount of sorting required.

You may use the information returned at the database manager level to help you tune the *sheapthres* configuration parameter. If the element value is greater than or equal to *sheapthres*, it means that the sorts are not getting the full sort heap as defined by the *sortheap* parameter.

Post Threshold Sorts

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Sort
Resettable	Yes	
Element Name	post_threshold_sorts	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Statement Sorts" on page 224 • "Active Sorts" on page 101	

Description: The number of sorts that have requested heaps after the sort heap threshold has been reached.

Usage: Under normal conditions, the database manager will allocate sort heap using the value specified by the *sortheap* configuration parameter. If the amount of memory allocated to sort heaps exceeds the sort heap threshold (*sheapthres* configuration parameter), the database manager will allocate sort heap using a value less than that specified by the *sortheap* configuration parameter.

Each active sort on the system allocates memory, which may result in sorting taking up too much of the system memory available. Sorts that start after the sort heap threshold has been reached may not receive an optimum amount of memory to execute, but, as a result, the entire system may benefit. By modifying the sort heap threshold and sort heap size configuration parameters, the performance of sort operations and/or the overall system can be improved. If this element's value is high, you can:

- Increase the sort heap threshold (*sheapthres*) or,
- Adjust applications to use fewer or smaller sorts via SQL query changes.

Piped Sorts Requested

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Resettable	Yes	
Element Name	piped_sorts_requested	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Piped Sorts Accepted" on page 97 • "Post Threshold Sorts" on page 96	

Description: The number of piped sorts that have been requested.

Usage: Each active sort on the system allocates memory, which may result in sorting taking up too much of the available system memory.

The sort list heap (*sortheap*) and sort heap threshold (*sheapthres*) configuration parameters help to control the amount of memory used for sort operations. These parameters are also used to determine whether a sort will be piped.

Since piped sorts may reduce disk I/O, allowing more piped sorts can improve the performance of sort operations and possibly the performance of the overall system. A piped sort is not be accepted if the sort heap threshold will be exceeded when the sort heap is allocated for the sort. See *Piped Sorts Accepted* for more information if you are experiencing piped sort rejections.

The SQL EXPLAIN output will show whether the optimizer requests a piped sort. For more information on piped and non-piped sorts see the *Administration Guide*.

Piped Sorts Accepted

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Resettable	Yes	
Element Name	piped_sorts_accepted	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Piped Sorts Requested" on page 97 • "Post Threshold Sorts" on page 96	

Description: The number of piped sorts that have been accepted.

Usage: Each active sort on the system allocates memory, which may result in sorting taking up too much of the available system memory.

When the number of accepted piped sorts is low compared to the number requested, you can improve sort performance by adjusting one or both of the following configuration parameters:

- sortheap
- sheapthres

If piped sorts are being rejected, you might consider decreasing your sort heap or increasing your sort heap threshold. You should be aware of the possible implications of either of these options. If you increase the sort heap threshold, then there is the possibility that more memory will remain allocated for sorting. This could cause the paging of memory to disk. If you decrease the sort heap, you might require an extra merge phase that could slow down the sort.

See the *Administration Guide* for more information on sorts.

Total Sorts

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Sort
Application	appl	Sort
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Statement	stmt_event	
Element Name	total_sorts	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Sort Overflows" on page 100 • "Total Sort Time" on page 99	

Description: The total number of sorts that have been executed.

Usage: At a database or application level, use this value with *Sort Overflows* to calculate the percentage of sorts that need more heap space. You can also use it with *Total Sort Time* to calculate the average sort time.

If the number of sort overflows is small with respect to the total sorts, then increasing the sort heap size may have little impact on performance, unless this buffer size is increased substantially.

At a statement level, use this element to identify statements which are performing large numbers of sorts. These statements may benefit from additional tuning to reduce the number of sorts. You can also use the SQL EXPLAIN statement to identify the number of sorts a statement performs. See the *Administration Guide* for more information.

Total Sort Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Sort
Application	appl	Sort
	stmt	Sort
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Statement	stmt_event	
Element Name	total_sort_time	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Total Sorts" on page 98 • "Total Sorts" on page 98	

Description: The total elapsed time (in milliseconds) for all sorts that have been executed.

Usage: At a database or application level, use this element with *Total Sorts* to calculate the average sort time, which can indicate whether or not sorting is an issue as far as performance is concerned.

At a statement level, use this element to identify statements that spend a lot of time sorting. These statements may benefit from additional tuning to reduce the sort time.

This count also includes sort time of temporary tables created during related operations. It provides information for one statement, one application, or all applications accessing one database.

When using data elements providing elapsed times, you should consider:

1. Elapsed times are affected by system load, so the more processes you have running, the higher this elapsed time value.
2. To calculate this data element at a database level, the database system monitor sums the application-level times. This can result in double counting elapsed times at a database level, since more than one application process can be running at the same time.

   To provide meaningful data from the database level, you should normalize the data to a lower level. For example:

   ```
   total sort time / total sorts
   ```

   provides information about the average elapsed time for each sort.

Sort Overflows

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Sort
Application	appl	Sort
	stmt	Sort
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Statement	stmt_event	
Element Name	sort_overflows	
Element Type	counter	
Related Information	• “Resetting Monitor Data” on page 25 • “When Counters are Initialized” on page 24 • “Total Sorts” on page 98	

Description: The total number of sorts that ran out of sort heap and may have required disk space for temporary storage.

Usage: At a database or application level, use this element in conjunction with *Total Sorts* to calculate the percentage of sorts that had to overflow to disk. If this percentage is high, you may want adjust the database configuration by increasing the value of *sortheap*.

At a statement level, use this element to identify statements that require large sorts. These statements may benefit from additional tuning to reduce the amount of sorting required.

When a sort overflows, additional overhead will be incurred because the sort will require a merge phase and can potentially require more I/O, if data needs to be written to disk.

This element provides information for one statement, one application, or all applications accessing one database.

Active Sorts

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	active_sorts	
Element Type	counter	
Related Information	• "Total Sort Heap Allocated" on page 95 • "Total Sorts" on page 98	

Description: The number of sorts in the database that currently have a sort heap allocated.

Usage: Use this value in conjunction with *Total Sort Heap Allocated* to determine the average sort heap space used by each sort. If the *sortheap* configuration parameter is substantially larger than the average sort heap used, you may be able to lower the value of this parameter. (See the *Administration Guide* for more details.)

This value includes heaps for sorts of temporary tables that were created during relational operations.

Hash Join

Hash join is an additional option for the optimizer. A hash join will first compare *hash codes* before comparing predicates for tables involved in a join. In a hash join, one table (selected by the optimizer) is scanned and rows are copied into memory buffers drawn from the sort heap allocation. The memory buffers are divided into partitions based on a hash code computed from the columns of the join predicates. Rows of the other table involved in the join are matched to rows from the first table by comparing the hash code. If the hash codes match, the actual join predicate columns are compared.

- "Total Hash Joins" on page 102
- "Hash Join Threshold" on page 102
- "Total Hash Loops" on page 103
- "Hash Join Overflows" on page 103

- "Hash Join Small Overflows" on page 104

Total Hash Joins

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	total_hash_joins	
Element Type	counter	
Related Information	• "Hash Join Threshold" on page 102 • "Total Hash Loops" on page 103 • "Hash Join Overflows" on page 103 • "Hash Join Small Overflows" on page 104	

Description: The total number of hash joins executed.

Usage: At the database or application level, use this value in conjunction with Hash Join Overflows and Hash Join Small Overflows to determine if a significant percentage of hash joins would benefit from modest increases in the sort heap size.

Hash Join Threshold

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Resettable	Yes	
Element Name	post_threshold_hash_joins	
Element Type	counter	
Related Information	• "Total Hash Joins" on page 102 • "Total Hash Loops" on page 103 • "Hash Join Overflows" on page 103 • "Hash Join Small Overflows" on page 104	

Description: The total number of times that a hash join heap request was limited due to concurrent use of shared or private sort heap space.

Usage: If this value is large (greater than 5% of "Hash Join Overflows" on page 103), the sort heap threshold should be increased.

Total Hash Loops

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	total_hash_loops	
Element Type	counter	
Related Information	• "Total Hash Joins" on page 102 • "Hash Join Threshold" on page 102 • "Hash Join Overflows" on page 103 • "Hash Join Small Overflows" on page 104	

Description: The total number of times that a single partition of a hash join was larger than the available sort heap space.

Usage: Values for this data element indicate inefficient execution of hash joins. This might indicate that the sort heap size is too small or the sort heap threshold is too small. Use this value in conjunction with the other hash join variables to tune the sort heap size (*sortheap*) and sort heap threshold (*sheapthres*) configuration parameters.

Hash Join Overflows

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	hash_join_overflows	
Element Type	counter	
Related Information	• "Total Hash Joins" on page 102 • "Hash Join Threshold" on page 102 • "Total Hash Loops" on page 103 • "Hash Join Small Overflows" on page 104	

Description: The number of times that hash join data exceeded the available sort heap space.

Usage: At the database level, if the percentage of Hash Join Small Overflows is greater than 10% of this value, then you should consider increasing the sort heap size. Values at the application level can be used to evaluate hash join performance for individual applications.

Hash Join Small Overflows

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	hash_join_small_overflows	
Element Type	counter	
Related Information	• "Total Hash Joins" on page 102 • "Hash Join Threshold" on page 102 • "Total Hash Loops" on page 103 • "Hash Join Overflows" on page 103	

Description: The number of times that hash join data exceeded the available sort heap space by less than 10%.

Usage: If this value and Hash Join Overflows are high, then you should consider increasing the sort heap threshold. If this value is greater than 10% of Hash Join Overflows, then you should consider increasing the sort heap size.

Fast Communication Manager

The following database system monitor elements provide information about the Fast Communication Manager (FCM):

- "FCM Buffers Currently Free" on page 105
- "Minimum FCM Buffers Free" on page 105
- "Message Anchors Currently Free" on page 106
- "Minimum Message Anchors" on page 106
- "Connection Entries Currently Free" on page 106
- "Minimum Connection Entries" on page 107

- "Request Blocks Currently Free" on page 107
- "Minimum Request Blocks" on page 108
- "Number of Nodes" on page 108
- "Connection Status" on page 108
- "Total FCM Buffers Sent" on page 109
- "Total FCM Buffers Received" on page 110

FCM Buffers Currently Free

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm	Basic
Resettable	No	
Element Name	buff_free	
Element Type	gauge	
Related Information	• "Minimum FCM Buffers Free" on page 105	

Description: This element indicates the number of FCM buffers currently free.

Usage: Use the number of FCM buffers currently free in conjunction with the *fcm_num_buffers* configuration parameter to determine the current FCM buffer pool utilization. You can use this information to tune *fcm_num_buffers*.

Minimum FCM Buffers Free

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm	Basic
Resettable	No	
Element Name	buff_free_bottom	
Element Type	water mark	
Related Information	• "FCM Buffers Currently Free" on page 105	

Description: The lowest number of free FCM buffers reached during processing.

Usage: Use this element in conjunction with the *fcm_num_buffers* configuration parameter to determine the maximum FCM buffer pool utilization. If buff_free_bottom is low, you should increase *fcm_num_buffers* to ensure that operations do not run out of FCM buffers. If buff_free_bottom is high, you can decrease *fcm_num_buffers* to conserve system resources.

Message Anchors Currently Free

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm	Basic
Resettable	No	
Element Name	MA_free	
Element Type	gauge	
Related Information	• "Minimum Message Anchors" on page 106	

Description: This element indicates the number of message anchors currently free.

Usage: Use the number of message anchors currently free in conjunction with the *fcm_num_anchors* configuration parameter to determine the current message anchor utilization. You can use this information to tune *fcm_num_anchors*.

Minimum Message Anchors

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm	Basic
Resettable	No	
Element Name	MA_free_bottom	
Element Type	water mark	
Related Information	• "Message Anchors Currently Free" on page 106	

Description: The lowest number of free message anchors reached during processing.

Usage: Use this element in conjunction with the *fcm_num_anchors* configuration parameter to determine the maximum message anchors utilization. If MA_free_bottom is low, you should increase *fcm_num_anchors* to ensure that operations do not run out of message anchors. If MA_free_bottom is high, you can decrease *fcm_num_anchors* to conserve system resources.

Connection Entries Currently Free

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm	Basic
Resettable	No	
Element Name	CE_free	
Element Type	gauge	
Related Information	• "Minimum Connection Entries" on page 107	

Description: This element indicates the number of connection entries currently free.

Usage: Use the number of connection entries currently free in conjunction with the *fcm_num_connect* configuration parameter to determine the current connection entry utilization. You can use this information to tune *fcm_num_connect*.

Minimum Connection Entries

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm	Basic
Resettable	No	
Element Name	CE_free_bottom	
Element Type	water mark	
Related Information	• “Connection Entries Currently Free” on page 106	

Description: The lowest number of free connection entries reached during processing.

Usage: Use this element in conjunction with the *fcm_num_connect* configuration parameter to determine the maximum connection entry utilization. If CE_free_bottom is low, you should increase *fcm_num_connect* to ensure that operations do not run out of connection entries. If CE_free_bottom is high, you can decrease *fcm_num_connect* to conserve system resources.

Request Blocks Currently Free

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm	Basic
Resettable	No	
Element Name	RB_free	
Element Type	gauge	
Related Information	• “Request Blocks Currently Free” on page 107	

Description: This element indicates the number of request blocks currently free.

Usage: Use the number of request blocks currently free in conjunction with the *fcm_num_rqb* configuration parameter to determine the current request block utilization. You can use this information to tune *fcm_num_rqb*.

Minimum Request Blocks

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic
Resettable	No	
Element Name	RB_free_bottom	
Element Type	water mark	
Related Information	• "Request Blocks Currently Free" on page 107	

Description: The lowest number of free request blocks reached during processing.

Usage: Use this element in conjunction with the *fcm_num_rqb* configuration parameter to determine the maximum request block utilization. If RB_free_bottom is low, you should increase *fcm_num_rqb* to ensure that operations do not run out of request blocks. If RB_free_bottom is high, you can decrease *fcm_num_rqb* to conserve system resources.

Number of Nodes

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic
Resettable	No	
Element Name	number_nodes	
Element Type	information	
Related Information	• None	

Description: The number of nodes in the current configuration.

Usage: Use this element to determine the number of *fcm_node* structures that will be returned.

Connection Status

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm_node	Basic
Resettable	No	
Element Name	connection_status	
Element Type	information	
Related Information	• "Total FCM Buffers Sent" on page 109 • "Total FCM Buffers Received" on page 110	

Description: This element indicates the status of the communication connection status between the node issuing the GET SNAPSHOT command and other nodes listed in the *db2nodes.cfg* file.

Usage: The connection values are :

SQLM_FCM_CONNECT_INACTIVE	No current connection
SQLM_FCM_CONNECT_ACTIVE	Connection is active
SQLM_FCM_CONNECT_CONGESTED	Connection is congested

Two nodes can be active, but the communication connection between them will remain inactive, unless there is some communication between those nodes.

Total FCM Buffers Sent

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm_node	Basic
Resettable	No	
Element Name	total_buffers_sent	
Element Type	counter	
Related Information	• “Connection Status” on page 108 • “Total FCM Buffers Received” on page 110	

Description: The total number of FCM buffers that have been sent from the node issuing the GET SNAPSHOT command to the node identified by the *node_number* (see the *db2nodes.cfg* file).

Usage: You can use this element to measure the level of traffic between the current node and the remote node. If the total number of FCM buffers sent to this node is high, you may want to redistribute the database, or move tables to reduce the inter-node traffic.

Total FCM Buffers Received

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	fcm_node	Basic
Resettable	No	
Element Name	total_buffers_rcvd	
Element Type	counter	
Related Information	• "Connection Status" on page 108 • "Total FCM Buffers Sent" on page 109	

Description: The total number of FCM buffers received by the node issuing the GET SNAPSHOT command from the node identified by the *node_number* (see the *db2nodes.cfg* file).

Usage: You can use this element to measure the level of traffic between the current node and the remote node. If the total number of FCM buffers received from this node is high, you may want to redistribute the database, or move tables to reduce the inter-node traffic.

Database Configuration

The following elements provide information particularly helpful for database performance tuning.

Buffer Pool Activity

The database server reads and updates all data from a buffer pool. Data is copied from disk to a buffer pool as it is required by applications.

Pages are placed in a buffer pool:

- by the agent. This is synchronous I/O.
- by the I/O servers (prefetchers). This is asynchronous I/O.

Pages are written to disk from a buffer pool:

- by the agent, synchronously
- by page cleaners, asynchronously

If the server needs to read a page of data, and that page is already in the buffer pool, then the ability to access that page is much faster than if the page had to be read from disk. It is desirable to **hit** as many pages as possible in the buffer pool. Avoiding disk I/O is the main issue when trying to improve the performance of your server And so, proper configuration of the buffer pools are probably the most important consideration for performance tuning.

Buffer Pool Hit Ratio

The buffer pool hit ratio indicates the percentage of time that the database manager did not need to load a page from disk in order to service a page request. That is, the page was already in the buffer pool. The greater the buffer pool hit ratio, the lower the frequency of disk I/O.

The buffer pool hit ratio can be calculated as follows:

```
(1 - ((pool_data_p_reads + pool_index_p_reads) /
  (pool_data_l_reads + pool_index_l_reads))) * 100%
```

This calculation takes into account all of the pages (index and data) that are cached by the buffer pool.

For a large database, increasing the buffer pool size may have minimal effect on the buffer pool hit ratio. Its number of data pages may be so large, that the statistical chances of a hit are not improved increasing its size. But you might find that tuning the index buffer pool hit ratio achieves the desired result. This can be achieved using two methods:

1. Split the data and indices into two different buffer pools and tune them separately.
2. Use one bufferpool, but increase its size until the index hit ratio stops increasing. The index buffer pool hit ratio can be calculated as follows:

   ```
   (1 - ((pool_index_p_reads) / (pool_index_l_reads))) * 100%
   ```

The first method is often more effective, but because it requires indices and data to reside in different tablespaces, it may not be an option for existing databases. It also requires tuning two bufferpools instead of one, which can be a more difficult task, particularly when memory is constrained.

Prefetchers

You should also consider the impact that prefetchers may be having on the hit ratio. Prefetchers read data pages into the buffer pool anticipating their need by an application (asynchronously). In most situations, these pages are read just before they are needed (the desired case). However, prefetchers can cause unnecessary I/O by reading pages into the buffer pool that will not be used. For example, an application starts reading through a table. This is detected and prefetching starts, but the application fills an application buffer and stops reading. Meanwhile, prefetching has been done for a number of additional pages. I/O has occurred for pages that will not be used and the buffer pool is partially taken up with those pages.

Page Cleaners

Page cleaners monitor the buffer pool and asynchronously write pages to disk. Their goals are:

- Ensure that agents will always find free pages in the buffer pool. If an agent does not find free pages in the buffer pool, it must clean them itself, and the associated application will have a poorer response.
- Speed database recovery, if a system crash occurs. The more pages that have been written to disk, the smaller the number of log file records that must be processed to recover the database.

Although dirty pages are written out to disk, the pages are not removed from the buffer pool right away, unless the space is needed to read in new pages.

Note: Buffer pool information is typically gathered at a table space level, but the facilities of the database system monitor can roll this information up to the buffer pool and database levels. Depending on your type of analysis, you may need to examine this data at any or all of these levels.

The following elements provide information about buffer pool activity. For an overview how the database manager uses buffer pools, see the *Administration Guide*.

- "Buffer Pool Data Logical Reads" on page 113
- "Buffer Pool Data Physical Reads" on page 115
- "Buffer Pool Data Writes" on page 116
- "Buffer Pool Index Logical Reads" on page 118
- "Buffer Pool Index Physical Reads" on page 119
- "Buffer Pool Index Writes" on page 120
- "Total Buffer Pool Physical Read Time" on page 122
- "Total Buffer Pool Physical Write Time" on page 123
- "Database Files Closed" on page 124
- "Buffer Pool Asynchronous Data Reads" on page 125
- "Buffer Pool Asynchronous Data Writes" on page 126
- "Buffer Pool Asynchronous Index Writes" on page 127
- "Buffer Pool Asynchronous Index Reads" on page 128
- "Buffer Pool Asynchronous Read Time" on page 129
- "Buffer Pool Asynchronous Write Time" on page 130

- "Buffer Pool Asynchronous Read Requests" on page 131
- "Buffer Pool Log Space Cleaners Triggered" on page 132
- "Buffer Pool Victim Page Cleaners Triggered" on page 133
- "Buffer Pool Threshold Cleaners Triggered" on page 134
- "Buffer Pool Information" on page 134
- "Bufferpool Name" on page 135
- "Time Waited for Prefetch" on page 135

Buffer Pool Data Logical Reads

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Connection	conn_event	
Element Name	pool_data_l_reads	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Database Activation Timestamp" on page 46 • "Connection Request Start Timestamp" on page 70 • "Buffer Pool Data Physical Reads" on page 115 • "Buffer Pool Data Writes" on page 116 • "Buffer Pool Index Logical Reads" on page 118 • "Buffer Pool Index Physical Reads" on page 119	

Description: Indicates the number of logical read requests for data pages that have gone through the buffer pool.

Usage: This count includes accesses to data that is:

- Already in the buffer pool when the database manager needs to process the page

- Read into the buffer pool before the database manager can process the page.

In conjunction with *Buffer Pool Data Physical Reads,* you can calculate the data page hit ratio for the buffer pool using the following formula:

```
1 - (buffer pool data physical reads / buffer pool data logical reads)
```

In conjunction with Buffer Pool Data Physical Reads, Buffer Pool Index Physical Reads, and Buffer Pool Index Logical Reads, you can calculate the overall buffer pool hit ratio using the following formula:

```
1 - ((buffer pool data physical reads + buffer pool index physical reads)
    / (buffer pool data logical reads + buffer pool index logical reads))
```

Increasing buffer pool size will generally improve the hit ratio, but you will reach a point of diminishing return. Ideally, if you could allocate a buffer pool large enough to store your entire database, then once the system is up and running you would get a hit ratio of 100%. However, this is unrealistic in most cases. the significance of the hit ratio really depends on the size of your data, and the way it is accessed. A very large database where data is accessed evenly would have a poor hit ratio. There is little you can do with very large tables. In such case, you would focus your attention on smaller, frequently accessed tables, and on the indices. Perhaps, assigning them to an individual buffer pools, for which you can aim for higher hit ratios.

Buffer Pool Data Physical Reads

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Connection	conn_event	
Element Name	pool_data_p_reads	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Database Activation Timestamp" on page 46 • "Connection Request Start Timestamp" on page 70 • "Buffer Pool Data Logical Reads" on page 113 • "Buffer Pool Data Writes" on page 116 • "Buffer Pool Index Logical Reads" on page 118 • "Buffer Pool Index Physical Reads" on page 119 • "Buffer Pool Asynchronous Data Reads" on page 125	

Description: The number of read requests that required I/O to get data pages into the buffer pool.

Usage: See Buffer Pool Data Logical Reads and Buffer Pool Asynchronous Data Reads for information about how to use this element.

Buffer Pool Data Writes

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Connection	conn_event	
Element Name	pool_data_writes	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Database Activation Timestamp" on page 46 • "Connection Request Start Timestamp" on page 70 • "Buffer Pool Data Logical Reads" on page 113 • "Buffer Pool Data Physical Reads" on page 115 • "Total Buffer Pool Physical Write Time" on page 123 • "Buffer Pool Asynchronous Data Writes" on page 126	

Description: Indicates the number of times a buffer pool data page was physically written to disk.

Usage: If a buffer pool data page is written to disk for a high percentage of the Buffer Pool Data Physical Reads, you may be able to improve performance by increasing the number of buffer pool pages available for the database.

A buffer pool data page is written to disk for the following reasons:

- To free a page in the buffer pool so another page can be read
- To flush the buffer pool.

The system does not always write a page to make room for a new one. If the page has not been updated, it can simply be replaced. This replacement is not counted for this element.

The data page can be written by an asynchronous page-cleaner agent before the buffer pool space is required. These asynchronous page writes are included in the value of this element in addition to synchronous page writes (see Buffer Pool Asynchronous Data Writes).

When calculating this percentage, disregard the number of physical reads required to initially fill the buffer pool. To determine the number of pages written:
1. Run your application (to load the buffer)
2. Note the value of this element
3. Run your application again
4. Subtract the value recorded in step 2 from the new value of this element.

In order to prevent the buffer pool from being deallocated between the runnings of your application, you should either;
- activate the database with the ACTIVATE DATABASE command
- have an idle application connected to the database.

If all applications are updating the database, increasing the size of the buffer pool may not have much impact on performance since most of the buffer pool pages contain updated data, which must be written to disk. However, if the updated pages can be used by other units of work before being written out, the buffer pool can save a write and a read, which will improve your performance.

See the *Administration Guide* for more information about buffer pool size.

Buffer Pool Index Logical Reads

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	**Logical Data Grouping**
Database	db_event
Table Space	tablespace_event
Connection	conn_event

Element Name	pool_index_l_reads
Element Type	counter

Related Information

- "Resetting Monitor Data" on page 25
- "When Counters are Initialized" on page 24
- "Database Activation Timestamp" on page 46
- "Connection Request Start Timestamp" on page 70
- "Buffer Pool Index Physical Reads" on page 119
- "Buffer Pool Index Writes" on page 120
- "Buffer Pool Data Physical Reads" on page 115
- "Buffer Pool Data Writes" on page 116
- "Buffer Pool Data Logical Reads" on page 113

Description: Indicates the number of logical read requests for index pages that have gone through the buffer pool.

Usage: This count includes accesses to index pages that are:

- Already in the buffer pool when the database manager needs to process the page
- Read into the buffer pool before the database manager can process the page.

In conjunction with Buffer Pool Index Physical Reads, you can calculate the index page hit ratio for the buffer pool using one of the following:

```
 1 - (buffer pool index physical reads / buffer pool index logical reads)
```

To calculate the overall buffer pool hit ratio, see Buffer Pool Data Logical Reads.

If the hit ratio is low, increasing the number of buffer pool pages may improve performance. See the *Administration Guide* for more information about buffer pool size.

Buffer Pool Index Physical Reads

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Connection	conn_event	
Element Name	pool_index_p_reads	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Database Activation Timestamp" on page 46 • "Connection Request Start Timestamp" on page 70 • "Buffer Pool Index Logical Reads" on page 118 • "Buffer Pool Index Writes" on page 120 • "Buffer Pool Data Logical Reads" on page 113 • "Buffer Pool Data Physical Reads" on page 115	

Description: Indicates the number of physical read requests to get index pages into the buffer pool.

Usage: See Buffer Pool Index Logical Reads for information about how to use this element.

Buffer Pool Index Writes

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Table Space	tablespace_event
Connection	conn_event

Element Name	pool_index_writes
Element Type	counter

Related Information

- “Resetting Monitor Data” on page 25
- “When Counters are Initialized” on page 24
- “Database Activation Timestamp” on page 46
- “Connection Request Start Timestamp” on page 70
- “Buffer Pool Index Logical Reads” on page 118
- “Buffer Pool Index Physical Reads” on page 119
- “Buffer Pool Asynchronous Index Writes” on page 127

Description: Indicates the number of times a buffer pool index page was physically written to disk.

Usage: Like a data page, a buffer pool index page is written to disk for the following reasons:

- To free a page in the buffer pool so another page can be read
- To flush the buffer pool.

The system does not always write a page to make room for a new one. If the page has not been updated, it can simply be replaced. This replacement is not counted for this element.

The index page can be written by an asynchronous page-cleaner agent before the buffer pool space is required. These asynchronous index page writes are included in the value of this element in addition to synchronous index page writes (see Buffer Pool Asynchronous Index Writes).

If a buffer pool index page is written to disk for a high percentage of the *Buffer Pool Index Physical Reads*, you may be able to improve performance by increasing the number of buffer pool pages available for the database.

When calculating this percentage, disregard the number of physical reads required to initially fill the buffer pool. To determine the number of pages written:

1. Run your application (to load the buffer)
2. Note the value of this element
3. Run your application again
4. Subtract the value recorded in step 2 from the new value of this element.

In order to prevent the buffer pool from being deallocated between the runnings of your application, you should either:

- activate the database with the ACTIVATE DATABASE command
- have an idle application connected to the database.

If all applications are updating the database, increasing the size of the buffer pool may not have much impact on performance, since most of the pages contain updated data which must be written to disk.

See the *Administration Guide* for more information about buffer pool size.

Total Buffer Pool Physical Read Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	**Logical Data Grouping**
Database	db_event
Table Space	tablespace_event
Connection	conn_event

Element Name	pool_read_time
Element Type	counter

Related Information

- "Resetting Monitor Data" on page 25
- "When Counters are Initialized" on page 24
- "Buffer Pool Data Physical Reads" on page 115
- "Buffer Pool Index Physical Reads" on page 119
- "Database Activation Timestamp" on page 46
- "Connection Request Start Timestamp" on page 70
- "Buffer Pool Asynchronous Read Time" on page 129

Description: Provides the total amount of elapsed time spent processing read requests that caused data or index pages to be physically read from disk to buffer pool.

Usage: You can use this element with Buffer Pool Data Physical Reads and Buffer Pool Index Physical Reads to calculate the average page-read time. This average is important since it may indicate the presence of an I/O wait, which in turn may indicate that you should be moving data to a different device.

At the database and table space levels, this element includes the value of Buffer Pool Asynchronous Read Time.

Total Buffer Pool Physical Write Time

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Poo
	bp_info	Buffer Pooll
Application	appl	Buffer Pool

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Table Space	tablespace_event
Connection	conn_event

Element Name	pool_write_time
Element Type	counter

Related Information

- "Resetting Monitor Data" on page 25
- "When Counters are Initialized" on page 24
- "Buffer Pool Data Writes" on page 116
- "Buffer Pool Index Writes" on page 120
- "Database Activation Timestamp" on page 46
- "Connection Request Start Timestamp" on page 70

Description: Provides the total amount of time spent physically writing data or index pages from the buffer pool to disk.

Usage: You can use this element with Buffer Pool Data Writes and Buffer Pool Index Writes to calculate the average page-write time. This average is important since it may indicate the presence of an I/O wait, which in turn may indicate that you should be moving data to a different device.

At the database and table space levels, this element includes the value of Buffer Pool Asynchronous Write Time.

Database Files Closed

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Element Name	files_closed	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24	

Description: The total number of database files closed.

Usage: The database manager opens files for reading and writing into and out of the buffer pool. The maximum number of database files open by an application at any time is controlled by the *maxfilop* configuration parameter. If the maximum is reached, one file will be closed before the new file is opened. Note that the actual number of files opened may not equal the number of files closed.

You can use this element to help you determine the best value for the *maxfilop* configuration parameter (see the *Administration Guide* for more information).

Buffer Pool Asynchronous Data Reads

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool

Resettable	Yes

Event Type	**Logical Data Grouping**
Database	db_event
Table Space	tablespace_event

Element Name	pool_async_data_reads
Element Type	counter

Related Information

- “Resetting Monitor Data” on page 25
- “When Counters are Initialized” on page 24
- “Buffer Pool Asynchronous Read Time” on page 129
- “Buffer Pool Data Physical Reads” on page 115
- “Buffer Pool Asynchronous Read Requests” on page 131
- “Direct Reads From Database” on page 141

Description: The number of pages read asynchronously into the buffer pool.

Usage: You can use this element with Buffer Pool Data Physical Reads to calculate the number of physical reads that were performed synchronously (that is, physical data page reads that were performed by database manager agents). Use the following formula:

```
buffer pool data physical reads - buffer pool asynchronous data reads
```

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the *num_ioservers* configuration parameter (see the *Administration Guide*).

Asynchronous reads are performed by database manager prefetchers. For information about these prefetchers, see the *Administration Guide*.

Buffer Pool Asynchronous Data Writes

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Table Space	tablespace_event

Element Name	pool_async_data_writes
Element Type	counter
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Asynchronous Index Writes" on page 127 • "Buffer Pool Data Writes" on page 116 • "Buffer Pool Asynchronous Write Time" on page 130 • "Buffer Pool Log Space Cleaners Triggered" on page 132 • "Buffer Pool Victim Page Cleaners Triggered" on page 133 • "Buffer Pool Threshold Cleaners Triggered" on page 134 • "Direct Writes to Database" on page 142

Description: The number of times a buffer pool data page was physically written to disk by either an asynchronous page cleaner, or a prefetcher. A prefetcher may have written dirty pages to disk to make space for the pages being prefetched.

Usage: You can use this element with Buffer Pool Data Writes to calculate the number of physical write requests that were performed synchronously (that is, physical data page writes that were performed by database manager agents). Use the following formula:

```
buffer pool data writes - buffer pool asynchronous data writes
```

By comparing the ratio of asynchronous to synchronous writes, you can gain insight into how well the buffer pool page cleaners are performing. This ratio can be helpful when you are tuning the *num_iocleaners* configuration parameter.

For more information about asynchronous page cleaners, see the *Administration Guide*.

Buffer Pool Asynchronous Index Writes

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool

Resettable	Yes

Event Type	**Logical Data Grouping**
Database	db_event
Table Space	tablespace_event

Element Name	pool_async_index_writes
Element Type	counter

Related Information

- "Resetting Monitor Data" on page 25
- "When Counters are Initialized" on page 24
- "Buffer Pool Asynchronous Data Writes" on page 126
- "Buffer Pool Asynchronous Index Reads" on page 128
- "Buffer Pool Index Writes" on page 120
- "Buffer Pool Asynchronous Write Time" on page 130
- "Buffer Pool Log Space Cleaners Triggered" on page 132
- "Buffer Pool Victim Page Cleaners Triggered" on page 133
- "Buffer Pool Threshold Cleaners Triggered" on page 134
- "Direct Writes to Database" on page 142

Description: The number of times a buffer pool index page was physically written to disk by either an asynchronous page cleaner, or a prefetcher. A prefetcher may have written dirty pages to disk to make space for the pages being prefetched.

Usage: You can use this element with Buffer Pool Index Writes to calculate the number of physical index write requests that were performed synchronously. That is, physical index page writes that were performed by database manager agents. Use the following formula:

```
buffer pool index writes - buffer pool asynchronous index writes
```

By comparing the ratio of asynchronous to synchronous writes, you can gain insight into how well the buffer pool page cleaners are performing. This ratio can be helpful when you are tuning the *num_iocleaners* configuration parameter.

For more information about asynchronous page cleaners, see the *Administration Guide*.

Buffer Pool Asynchronous Index Reads

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Element Name	pool_async_index_reads	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Asynchronous Data Writes" on page 126 • "Buffer Pool Asynchronous Index Writes" on page 127 • "Buffer Pool Index Physical Reads" on page 119 • "Buffer Pool Asynchronous Read Time" on page 129 • "Buffer Pool Log Space Cleaners Triggered" on page 132 • "Buffer Pool Victim Page Cleaners Triggered" on page 133 • "Buffer Pool Threshold Cleaners Triggered" on page 134 • "Direct Reads From Database" on page 141	

Description: The number of index pages read asynchronously into the buffer pool by a prefetcher.

Usage: You can use this element with Buffer Pool Index Physical Reads to calculate the number of physical reads that were performed synchronously (that is, physical index page reads that were performed by database manager agents). Use the following formula:

```
 buffer pool index physical reads - buffer pool asynchronous index reads
```

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the *num_ioservers* configuration parameter (see the *Administration Guide*).

Asynchronous reads are performed by database manager prefetchers. For information about these prefetchers, see the *Administration Guide*.

Buffer Pool Asynchronous Read Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Element Name	pool_async_read_time	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Asynchronous Data Reads" on page 125 • "Total Buffer Pool Physical Read Time" on page 122 • "Buffer Pool Asynchronous Read Requests" on page 131 • "Direct Read Time" on page 145	

Description: The total elapsed time spent reading by database manager prefetchers.

Usage: You can use this element to calculate the elapsed time for synchronous reading, using the following formula:

```
  total buffer pool physical read time - buffer pool asynchronous read time
```

You can also use this element to calculate the average asynchronous read time using the following formula:

```
  buffer pool asynchronous read time / buffer pool asynchronous data reads
```

These calculations can be used to understand the I/O work being performed.

Buffer Pool Asynchronous Write Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Element Name	pool_async_write_time	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Asynchronous Data Writes" on page 126 • "Buffer Pool Asynchronous Index Writes" on page 127 • "Total Buffer Pool Physical Write Time" on page 123 • "Buffer Pool Asynchronous Read Requests" on page 131 • "Direct Write Time" on page 146	

Description: The total elapsed time spent writing data or index pages from the buffer pool to disk by database manager page cleaners.

Usage: To calculate the elapsed time spent writing pages synchronously, use the following formula:

```
 total buffer pool physical write time - buffer pool asynchronous write time
```

You can also use this element to calculate the average asynchronous read time using the following formula:

```
 buffer pool asynchronous write time
  / (buffer pool asynchronous data writes
    + buffer pool asynchronous index writes)
```

These calculations can be used to understand the I/O work being performed.

Buffer Pool Asynchronous Read Requests

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Table Space	tablespace_event	
Element Name	pool_async_data_read_reqs	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Asynchronous Data Reads" on page 125	

Description: The number of asynchronous read requests.

Usage: To calculate the average number of data pages read per asynchronous request, use the following formula:

```
buffer pool asynchronous data reads / buffer pool asynchronous read requests
```

This average can help you determine the amount of asynchronous I/O done in each interaction with the prefetcher.

Buffer Pool Log Space Cleaners Triggered

Snapshot Level Database	**Logical Data Grouping** dbase	**Monitor Switch** Buffer Pool
Resettable	Yes	
Event Type Database	**Logical Data Grouping** db_event	
Element Name **Element Type**	pool_lsn_gap_clns counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Victim Page Cleaners Triggered" on page 133 • "Buffer Pool Threshold Cleaners Triggered" on page 134	

Description: The number of times a page cleaner was invoked because the logging space used had reached a predefined criterion for the database.

Usage: This element can be used to help evaluate whether you have enough space for logging, and whether you need more log files or larger log files.

The page cleaning criterion is determined by the setting for the *softmax* configuration parameter. Page cleaners are triggered if the oldest page in the buffer pool contains an update described by a log record that is older than the current log position by the criterion value. See the *Administration Guide* for more information.

Buffer Pool Victim Page Cleaners Triggered

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	pool_drty_pg_steal_clns	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Log Space Cleaners Triggered" on page 132 • "Buffer Pool Threshold Cleaners Triggered" on page 134	

Description: The number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the database.

Usage: Using the following formula, you may calculate what percentage of all cleaner invocations are represented by this element:

```
   buffer pool victim page cleaners triggered
 / ( buffer pool victim page cleaners triggered
   + buffer pool threshold cleaners triggered
   + buffer pool log space cleaners triggered)
```

If this ratio is low, it may indicate that you have defined too many page cleaners. If your *chngpgs_thresh* is set too low, you may be writing out pages that you will dirty later. Aggressive cleaning defeats one purpose of the buffer pool, that is to defer writing to the last possible moment.

If this ratio is high, it may indicate that you have too few page cleaners defined. Too few page cleaners will increase recovery time after failures (see the *Administration Guide*).

Note: Although dirty pages are written out to disk, the pages are not removed from the buffer pool right away, unless the space is needed to read in new pages.

Buffer Pool Threshold Cleaners Triggered

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	pool_drty_pg_thrsh_clns	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Buffer Pool Log Space Cleaners Triggered" on page 132	

Description: The number of times a page cleaner was invoked because a buffer pool had reached the dirty page threshold criterion for the database.

Usage: The threshold is set by the *chngpgs_thresh* configuration parameter. It is a percentage applied to the buffer pool size. When the number of dirty pages in the pool exceeds this value, the cleaners are triggered.

If this value is set too low, pages might be written out too early, requiring them to be read back in. If set too high, then too many pages may accumulate, requiring users to write out pages synchronously. See the *Administration Guide* for more information.

Buffer Pool Information

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table Space	bufferpool	Buffer Pool
Resettable	No	
Event Type	**Logical Data Grouping**	
Table Space	bufferpool_event	
Element Name	bp_info	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25	

Description: Data management counters for a buffer pool.

Usage: Activity performed for a buffer pool.

Bufferpool Name

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	bufferpool	Basic
Resettable	No	
Element Name	bp_name	
Element Type	information	
Related Information	• None	

Description: The name of the buffer pool.

Usage: A new database has a default buffer pool called IBMDEFAULTBP with a size determined by the platform. Each database requires at least one buffer pool. However, depending on your needs you may choose to create several buffer pools, each of a different size, for a single database. The CREATE, ALTER, and DROP BUFFERPOOL statements allow you to create, change, or remove a buffer pool.

Time Waited for Prefetch

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Application	appl	Buffer Pool
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	prefetch_wait_time	
Element Type	counter	
Related Information	• None	

Description: The time an application spent waiting for an I/O server (prefetcher) to finish loading pages into the buffer pool.

Usage: This element can be used to experiment with changing the number of I/O servers, and I/O server sizes.

Extended Storage

Extended storage provides a secondary level of storage for bufferpools. This allows a user to access memory beyond the maximum allowed for each process. Extended storage consists of segments that will be allocated in addition to the bufferpools. The extended storage will assign pages to segments that are attached or detached, as needed. The number and size of segments are configurable. Attachment is allowed to only one segment at a given time.

There is one extended storage for all buffer pools, and each buffer pool can be configured to use it or not. See the *Administration Guide* for more information.

Extended storage should only be used on systems with very large amount of real memory. These are systems that have more memory than can be attached to by a single process.

Using Extended Storage Counters: If you have extended storage set on for a buffer pool, all pages removed from the buffer pool will be written to extended storage. Each of these writes has a cost associated with it. Some of these pages may never be required or they may be forced out of extended storage before they are ever read back into the buffer pool.

You can calculate the extended storage read/write ratio as follows:

```
(data + index copied from extended storage)
/ (data + index copied to extended storage)
```

Where the numerator in this equation is pages from extended storage to buffer pool and the denominator is pages from bufferpool to extended storage.

The top portion of this equation represents a performance saving. When a page is transferred from extended storage to buffer pool, you save a system I/O call. However, you still incur the cost of attaching to the extended memory segment, copying the page, and detaching from the segment. The bottom part represents the cost of transferring a page to extended storage, that is, attaching to the segment, copying the page, and detaching.

The higher the ratio, the more likely you are to benefit from extended storage. In general, extended storage is particularly useful if I/O activity is very high on your system.

There is a crossover point where the cost of copying pages to be removed from the buffer pool to extended storage equals the savings from reading pages from extended storage, instead of having to read them from disk. This crossover point is affected by:

- cost of an I/O on your system
- cost of copying data in memory and accessing shared memory segments

It is difficult to establish an exact crossover point. To establish a baseline, you must experiment by enabling extended storage for different buffer pools, and determine whether it improves your overall database performance. This can be measured by using application benchmarks. For instance, you may want to monitor transaction rates and execution time. See the *Administration Guide* for information on benchmarking.

Once you have established that extended storage is beneficial for some buffer pools. You want to measure the read / write ratio to obtain a baseline. This ratio is most important during database creation and initial setup. After that, you want to monitor this ratio to ensure that it is not deviating from the initial baseline.

The following elements provide information about buffer pools and extended storage. For more information on how the database manager uses extended storage, see the *Administration Guide*.

- "Buffer Pool Data Pages to Extended Storage"
- "Buffer Pool Index Pages to Extended Storage" on page 138
- "Buffer Pool Data Pages from Extended Storage" on page 139
- "Buffer Pool Index Pages from Extended Storage" on page 140

Buffer Pool Data Pages to Extended Storage

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Table Space	tablespace_event	
Element Name	pool_data_to_estore	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "Buffer Pool Index Pages to Extended Storage" on page 138 • "Buffer Pool Data Pages from Extended Storage" on page 139 • "Buffer Pool Index Pages from Extended Storage" on page 140	

Description: Number of buffer pool data pages copied to extended storage.

Usage: Pages are copied from the buffer pool to extended storage, when they are selected as victim pages. This copying is required to make space for new pages in the buffer pool.

Buffer Pool Index Pages to Extended Storage

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Table Space	tablespace_event	
Element Name	pool_index_to_estore	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "Buffer Pool Data Pages to Extended Storage" on page 137 • "Buffer Pool Data Pages from Extended Storage" on page 139 • "Buffer Pool Index Pages from Extended Storage" on page 140	

Description: Number of buffer pool index pages copied to extended storage.

Usage: Pages are copied from the buffer pool to extended storage, when they are selected as victim pages. This copying is required to make space for new pages in the buffer pool.

Buffer Pool Data Pages from Extended Storage

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	**Logical Data Grouping**
Database	db_event
Connection	conn_event
Table Space	tablespace_event

Element Name	pool_data_from_estore
Element Type	counter

Related Information

- “Resetting Monitor Data” on page 25
- “Buffer Pool Data Pages to Extended Storage” on page 137
- “Buffer Pool Index Pages to Extended Storage” on page 138
- “Buffer Pool Index Pages from Extended Storage” on page 140

Description: Number of buffer pool data pages copied from extended storage.

Usage: Required pages are copied from extended storage to the buffer pool, if they are not in the buffer pool, but are in extended storage. This copying may incur the cost of connecting to the shared memory segment, but saves the cost of a disk read.

Buffer Pool Index Pages from Extended Storage

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Table Space	tablespace_event	
Element Name	pool_index_from_estore	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "Buffer Pool Data Pages to Extended Storage" on page 137 • "Buffer Pool Index Pages to Extended Storage" on page 138 • "Buffer Pool Data Pages from Extended Storage" on page 139	

Description: Number of buffer pool index pages copied from extended storage.

Usage: Required index pages are copied from extended storage to the buffer pool, if they are not in the buffer pool, but are in extended storage. This copying may incur the cost of connecting to the shared memory segment, but saves the cost of a disk read.

Non-buffered I/O Activity

The following elements provide information about I/O activity that does not use the buffer pool:

- "Direct Reads From Database" on page 141
- "Direct Writes to Database" on page 142
- "Direct Read Requests" on page 143
- "Direct Write Requests" on page 144
- "Direct Read Time" on page 145
- "Direct Write Time" on page 146

Direct Reads From Database

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Table Space	tablespace_event	
Element Name	direct_reads	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Direct Read Requests" on page 143 • "Direct Read Time" on page 145 • "Direct Writes to Database" on page 142	

Description: The number of read operations that do not use the buffer pool.

Usage: Use the following formula to calculate the average number of sectors that are read by a direct read:

```
direct reads from database / direct read requests
```

When using system monitors to track I/O, this data element helps you distinguish database I/O from non-database I/O on the device.

Direct reads are performed in units, the smallest being a 512-byte sector. They are used when:

- Reading LONG VARCHAR columns
- Reading LOB (large object) columns
- Performing a backup

Direct Writes to Database

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Table Space	tablespace_event	
Element Name	direct_writes	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Direct Write Requests" on page 144 • "Direct Write Time" on page 146 • "Direct Reads From Database" on page 141	

Description: The number of write operations that do not use the buffer pool.

Usage: Use the following formula to calculate the average number of sectors that are written by a direct write.

```
direct writes to database / direct write requests
```

When using system monitors to track I/O, this data element helps you distinguish database I/O from non-database I/O on the device.

Direct writes are performed in units, the smallest being a 512-byte sector. They are used when:

- Writing LONG VARCHAR columns
- Writing LOB (large object) columns
- Performing a restore
- Performing a load.

Direct Read Requests

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event
Table Space	tablespace_event

Element Name	direct_read_reqs
Element Type	counter

Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Direct Reads From Database" on page 141 • "Direct Read Time" on page 145 • "Direct Write Requests" on page 144

Description: The number of requests to perform a direct read of one or more sectors of data.

Usage: Use the following formula to calculate the average number of sectors that are read by a direct read:

```
direct reads from database / direct read requests
```

Direct Write Requests

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event
Table Space	tablespace_event

Element Name	direct_write_reqs
Element Type	counter

Related Information

- "Resetting Monitor Data" on page 25
- "When Counters are Initialized" on page 24
- "Direct Writes to Database" on page 142
- "Direct Write Time" on page 146
- "Direct Read Requests" on page 143

Description: The number of requests to perform a direct write of one or more sectors of data.

Usage: Use the following formula to calculate the average number of sectors that are written by a direct write:

```
direct writes to database / direct write requests
```

Direct Read Time

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event
Table Space	tablespace_event

Element Name	direct_read_time
Element Type	counter
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Direct Reads From Database" on page 141 • "Direct Read Requests" on page 143 • "Direct Write Time" on page 146

Description: The elapsed time (in milliseconds) required to perform the direct reads.

Usage: Use the following formula to calculate the average direct read time per sector:

```
direct read time / direct reads from database
```

A high average time may indicate an I/O conflict.

Direct Write Time

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
	bp_info	Buffer Pool
Application	appl	Buffer Pool

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event
Table Space	tablespace_event

Element Name	direct_write_time
Element Type	counter
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Direct Writes to Database" on page 142 • "Direct Write Requests" on page 144 • "Direct Read Time" on page 145

Description: The elapsed time (in milliseconds) required to perform the direct writes.

Usage: Use the following formula to calculate the average direct write time per sector:

```
direct write time / direct writes to database
```

A high average time may indicate an I/O conflict.

Catalog Cache

The catalog cache stores table descriptors for tables, views, and aliases. A descriptor stores information about a table, view, or alias in a condensed internal format. When a transaction references a table, it causes an insert of a table descriptor into the cache, so that subsequent transactions referencing that same table can use that descriptor and avoid reading from disk. (Transactions reference a table descriptor when compiling an SQL statement.)

The following database system monitor elements are used for catalog caches:

- "Catalog Cache Lookups" on page 147
- "Catalog Cache Inserts" on page 148

- “Catalog Cache Overflows” on page 148
- “Catalog Cache Heap Full” on page 149

Catalog Cache Lookups

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	cat_cache_lookups	
Element Type	counter	
Related Information	• “Catalog Cache Inserts” on page 148 • “Catalog Cache Overflows” on page 148 • “Catalog Cache Heap Full” on page 149	

Description: The number of times that the catalog cache was referenced to obtain table descriptor information.

Usage: This element includes both successful and unsuccessful accesses to the catalog cache. The catalog cache is referenced whenever a table, view, or alias name is processed during the compilation of an SQL statement.

To calculate the catalog cache hit ratio use the following formula:

```
  (1 - (cat_cache_inserts / cat_cache_lookups))
```

indicates how well the catalog cache is avoiding catalog accesses. If the ratio is high (more than 0.8), then the cache is performing well. A smaller ratio might suggest that the *catalogcache_sz* should be increased. You should expect a large ratio immediately following the first connection to the database.

The execution of Data Definition Language (DDL) SQL statements involving a table, view, or alias will evict the table descriptor information for that object from the catalog cache causing it to be re-inserted on the next reference. Therefore, the heavy use of DDLs may also increase the ratio.

See the *Administration Guide* for more information on the Catalog Cache Size configuration parameter.

Catalog Cache Inserts

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	cat_cache_inserts	
Element Type	counter	
Related Information	• "Catalog Cache Lookups" on page 147 • "Catalog Cache Overflows" on page 148 • "Catalog Cache Heap Full" on page 149 • "Data Definition Language (DDL) SQL Statements" on page 208	

Description: The number of times that the system tried to insert table descriptor information into the catalog cache.

Usage: Table descriptor information is usually inserted into the cache following a failed lookup to the catalog cache while processing a table, view, or alias reference in an SQL statement. The *catalog cache inserts* value includes attempts to insert table descriptor information that fail due to catalog cache overflow and heap full conditions.

See "Catalog Cache Lookups" on page 147 for more catalog cache information.

Catalog Cache Overflows

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	cat_cache_overflows	
Element Type	counter	
Related Information	• "Catalog Cache Lookups" on page 147 • "Catalog Cache Inserts" on page 148	

Description: The number of times that an insert into the catalog cache failed due the catalog cache being full.

Usage: The catalog cache space is filled with table descriptor information.

The cache entries for transactions that compile SQL statements, either by issuing dynamic SQL statements or by binding a package, will not be eligible to be removed from the cache until that transaction has either been committed or rolled back. Catalog cache space is reclaimed by evicting table descriptor information for tables, views, or aliases that are not currently in use by any transaction. Once a transaction has experienced a catalog cache overflow, all subsequent attempts by the same transaction to insert table descriptor information into the catalog cache will also result in an overflow.

Note: A transaction involved in an overflow will proceed, but its descriptor information is not inserted into the cache.

If *catalog cache overflows* is large, the catalog cache may be too small for the workload. Enlarging the catalog cache may improve its performance. If the workload includes transactions which compile a large number of SQL statements referencing many tables, views, and aliases in a single unit of work, then compiling fewer SQL statements in a single transaction may improve the performance of the catalog cache. Or if it includes binding of packages containing many SQL statements referencing many tables, views or aliases, you can try splitting packages so that they include fewer SQL statements to improve performance.

Catalog Cache Heap Full

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	cat_cache_heap_full	
Element Type	counter	
Related Information	• “Package Cache Inserts” on page 153 • “Data Definition Language (DDL) SQL Statements” on page 208 • “Dynamic SQL Statements Attempted” on page 203 • “Static SQL Statements Attempted” on page 203	

Description: The number of times that an insert into the catalog cache failed due to a heap-full condition in the database heap.

Usage: The catalog cache draws its storage dynamically from the database heap and even if the cache storage has not reached its limit, inserts into the catalog cache may fail due to a lack of space in the database heap.

If the catalog cache heap full count is not zero, then this insert failure condition can be corrected by increasing the database heap size or reducing the catalog cache size.

Package Cache

The package and section information required for the execution of dynamic and static SQL statements are placed in the package cache as required. This information is required whenever a dynamic or static statement is being executed. The package cache exists at a database level. This means that agents with similar environments can share the benefits of another agent's work. For static SQL statements, this can mean avoiding catalog access. For dynamic SQL statements, this can mean avoiding the cost of compilation.

The following database system monitor elements are used for package caches:

- "Package Cache Lookups" on page 151
- "Package Cache Inserts" on page 153
- "Package Cache Overflows" on page 153
- "Maximum Package Cache Size" on page 154
- "Section Lookups" on page 155
- "Section Inserts" on page 156

Package Cache Lookups

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event

Element Name	pkg_cache_lookups
Element Type	counter

Related Information

- "Package Cache Inserts" on page 153
- "Section Lookups" on page 155
- "Section Inserts" on page 156
- "Static SQL Statements Attempted" on page 203
- "Dynamic SQL Statements Attempted" on page 203
- "Data Definition Language (DDL) SQL Statements" on page 208

Description: The number of times that an application looked for a section or package in the package cache. At a database level, it indicates the overall number of references since the database was started, or monitor data was reset.

Note: This counter includes the cases where the section is already loaded in the cache and when the section has to be loaded into the cache.

Usage: To calculate the package cache hit ratio use the following formula:

```
   1 - (Package Cache Inserts / Package Cache Lookups)
```

The package cache hit ratio tells you whether or not the package cache is being used effectively. If the hit ratio is high (more than 0.8), the cache is performing well. A smaller ratio may indicate that the package cache should be increased.

You will need to experiment with the size of the package cache to find the optimal number for the *pckcachesz* configuration parameter. For example, you might be able to use a smaller package cache size if there is no increase in the *pkg_cache_inserts* data element when you decrease the size of the cache. Decreasing the package cache size frees up system resources for other work. It is also possible that you could improve overall system performance by increasing the size of the package cache if by doing so, you decrease the number of package cache inserts. This experimentation is best done under full workload conditions.

You can use this data element with *ddl_sql_stmts* to determine whether or not the execution of DDL statements is impacting the performance of the package cache. Sections for dynamic SQL statements can become invalid when DDL statements are executed. Invalid sections are implicitly prepared by the system when next used. The execution of a DDL statement could invalidate a number of sections and the resulting extra overhead incurred when preparing those sections could significantly impact performance. In this case, the package cache hit ratio reflects the implicit recompilation of invalid sections and not the insertion of new sections into the cache, so increasing the size of the package cache will not improve overall performance. You might find it less confusing to tune the cache for an application on its own before working in the full environment.

It is necessary to determine the role that DDL statements are playing in the value of the package cache hit ratio before deciding on what action to take. If DDL statements rarely occur, then cache performance may be improved by increasing its size. If DDL statements are frequent, then improvements may require that you limit the use of DDL statements (possibly to specific time periods).

The *static_sql_stmts* and *dynamic_sql_stmts* counts can be used to help provide information on the quantity and type of sections being cached.

See the *Administration Guide* for more information on the Package Cache Size (*pckcachesz*) configuration parameter.

Note: You may want to use this information at the database level to calculate the average package cache hit ratio all each applications. You should look at this information at an application level to find out the exact package cache hit ratio for a given application. It may not be worthwhile to increase the size of the package cache in order to satisfy the cache requirements of an application that only executes infrequently.

Package Cache Inserts

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event

Element Name	pkg_cache_inserts
Element Type	counter

Related Information	• “Package Cache Lookups” on page 151 • “Section Lookups” on page 155 • “Section Inserts” on page 156

Description: The total number of times that a requested section was not available for use and had to be loaded into the package cache. This count includes any implicit prepares performed by the system.

Usage: In conjunction with ″Package Cache Lookups″, you can calculate the package cache hit ratio using the following formula:

```
1 - (Package Cache Inserts / Package Cache Lookups)
```

See “Package Cache Lookups” on page 151 for information on using this element.

Package Cache Overflows

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event

Element Name	pkg_cache_num_overflows
Element Type	counter

Related Information	• “Package Cache Inserts” on page 153 • “Data Definition Language (DDL) SQL Statements” on page 208 • “Dynamic SQL Statements Attempted” on page 203 • “Static SQL Statements Attempted” on page 203

Description: The number of times that the package cache overflowed the bounds of its allocated memory.

Usage: Use this element with pkg_cache_size_top to determine whether the size of the package cache needs to be increased to avoid overflowing. Overflows of the package cache can cause unnecessary lock escalations, resulting in loss of concurrency, or out of memory errors from the other heaps allocated out of the database shared memory, as well as performance degradation.

Maximum Package Cache Size

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	pkg_cache_size_top	
Element Type	water mark	
Related Information	• "Package Cache Overflows" on page 153	

Description: The largest size reached by the package cache.

Usage: This element indicates the maximum number of bytes the package cache required for the workload run against the database since it was activated.

If the package cache overflowed, then this element contains the largest size reached by the package cache during the overflow. Check Package Cache Overflows to determine if such a condition occurred.

When the package cache overflows, memory is temporarily borrowed from other entities in database shared memory (for example, lock list or database heap). This can result in memory shortage errors from these entities or performance degradation from concurrency reduction due to unnecessary lock escalations. You can determine the minimum size of the package cache required by your workload by:

```
maximum package cache size / 4096
```

Rounding the result up to a whole number, indicates the minimum number of 4K pages required by the package cache to avoid overflow.

Section Lookups

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event

Element Name	appl_section_lookups
Element Type	counter
Related Information	• "Package Cache Lookups" on page 151 • "Package Cache Inserts" on page 153 • "Section Inserts" on page 156

Description: Lookups of SQL sections by an application from its SQL work area.

Usage: Each agent has access to a unique SQL work area where the working copy of any executable section is kept. In partitioned databases, this work area is shared by all non-SMP agents. In other environments and with SMP agents, each agent has its own unique SQL work area.

This counter indicates how many times the SQL work area was accessed by agents for an application. It is a cumulative total of all lookups on all SQL work heaps for agents working for this application.

You can use this element in conjunction with *"Section Inserts" on page 156* to tune the size of the heap used for the SQL work area. In partitioned databases this size is controlled by the *app_ctl_heap_sz* configuration parameter. SQL work area size in other database environments uses the the *applheapsz* configuration parameter. The size of the SQL work area for SMP agents is controlled by *applheapsz* in all environments.

Section Inserts

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	appl_section_inserts	
Element Type	counter	
Related Information	• "Package Cache Lookups" on page 151 • "Package Cache Inserts" on page 153 • "Section Lookups" on page 155	

Description: Inserts of SQL sections by an application from its SQL work area.

Usage: The working copy of any executable section is stored in a unique SQL work area. This is a count of when a copy was not available and had to be inserted. See "Section Lookups" on page 155 for more information on using sections.

Database Heap

The following database system monitor elements are used for database heaps:
- "Maximum Database Heap Allocated"

Maximum Database Heap Allocated

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	db_heap_top	
Element Type	water mark	
Related Information	• None	

Description: The largest amount of database heap allocated and used by the database, since the first application connected to the database (in bytes).

Usage: You may use this element to evaluate the setting of the *dbheap* configuration parameter, which is described in the *Administration Guide*. The *dbheap* parameter limits the amount of storage that can be allocated for database heap.

If the value of this element is the same as the *dbheap* parameter, it is quite likely that an application has received an error indicating that there was not enough storage available.

Logging

The following database system monitor elements are used only when circular logging is being used. That is, they are not used if either the *logretain* or *userexit* configuration parameter is enabled.

- "Maximum Secondary Log Space Used" on page 158
- "Maximum Total Log Space Used" on page 159
- "Secondary Logs Allocated Currently" on page 160

The following database system monitor elements are used for all types of logging:

- "Number of Log Pages Read" on page 160
- "Number of Log Pages Written" on page 161
- "Unit of Work Log Space Used" on page 162
- "Total Log Space Used" on page 162
- "Total Log Available" on page 163

For more information about logging and log configuration parameters, see the *Administration Guide.*

Maximum Secondary Log Space Used

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	sec_log_used_top	
Element Type	water mark	
Related Information	• "Unit of Work Log Space Used" on page 162 • "Secondary Logs Allocated Currently" on page 160 • "Maximum Total Log Space Used" on page 159	

Description: The maximum amount of secondary log space used (in bytes).

Usage: You may use this element in conjunction with *Secondary Logs Allocated Currently* and *Maximum Total Log Space Used* to show your current dependency on secondary logs. If this value is high, you may need larger log files, or more primary log files, or more frequent COMMIT statements within your application.

As a result, you may need to adjust the following configuration parameters:

- logfilsz
- logprimary
- logsecond
- logretain

The value will be zero if the database does not have any secondary log files. This would be the case if there were none defined.

For more information, see the *Administration Guide*.

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Maximum Total Log Space Used

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	tot_log_used_top	
Element Type	water mark	
Related Information	• “Unit of Work Log Space Used” on page 162 • “Secondary Logs Allocated Currently” on page 160 • “Maximum Secondary Log Space Used” on page 158	

Description: The maximum amount of total log space used (in bytes).

Usage: You can use this element to help you evaluate the amount of primary log space that you have allocated. Comparing the value of this element with the amount of primary log space you have allocated can help you to evaluate your configuration parameter settings. Your primary log space allocation can be calculated using the following formula:

```
logprimary x logfilsiz x 4096 (see note below)
```

You can use this element in conjunction with *Maximum Secondary Log Space Used* and *Secondary Logs Allocated Currently* to show your current dependency on secondary logs.

This value includes space used in both primary and secondary log files, and is only returned if circular logging is used. (That is, it is not returned if either the *logretain* or *userexit* configuration parameter is enabled.)

As a result, you may need to adjust the following configuration parameters:

- logfilsz
- logprimary
- logsecond
- logretain

For more information, see the *Administration Guide*.

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Secondary Logs Allocated Currently

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	sec_logs_allocated	
Element Type	gauge	
Related Information	• "Unit of Work Log Space Used" on page 162 • "Maximum Secondary Log Space Used" on page 158 • "Maximum Total Log Space Used" on page 159	

Description: The total number of secondary log files that are currently being used for the database.

Usage: You may use this element in conjunction with *Maximum Secondary Log Space Used* and *Maximum Total Log Space Used* to show your current dependency on secondary logs. If this value is consistently high, you may need larger log files, or more primary log files, or more frequent COMMIT statements within your application.

As a result, you may need to adjust the following configuration parameters:

- logfilsz
- logprimary
- logsecond
- logretain

For more information, see the *Administration Guide*.

Number of Log Pages Read

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Element Name	log_reads	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Number of Log Pages Written" on page 161	

Description: The number of log pages read from disk by the logger.

Usage: You can use this element with an operating system monitor to quantify the amount of I/O on a device that is attributable to database activity.

Number of Log Pages Written

Snapshot Level Database	**Logical Data Grouping** dbase	**Monitor Switch** Basic
Resettable	Yes	
Event Type Database	**Logical Data Grouping** db_event	
Element Name **Element Type**	log_writes counter	
Related Information	• “When Counters are Initialized” on page 24 • “Number of Log Pages Read” on page 160	

Description: The number of log pages written to disk by the logger.

Usage: You may use this element with an operating system monitor to quantify the amount of I/O on a device that is attributable to database activity.

Note: When log pages are written to disk, the last page might not be full. In such cases, the partial log page remains in the log buffer, and additional log records are written to the page. Therefore log pages might be written to disk by the logger more than once. You should not use this data element to measure the number of pages produced by DB2.

Unit of Work Log Space Used

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Unit of Work
Resettable	No	
Event Type	**Logical Data Grouping**	
Transaction	xaction_event	
Element Name	uow_log_space_used	
Element Type	gauge	
Related Information	• "Resetting Monitor Data" on page 25 • "Secondary Logs Allocated Currently" on page 160 • "Maximum Secondary Log Space Used" on page 158 • "Maximum Total Log Space Used" on page 159	

Description: The amount of log space (in bytes) used in the current unit of work of the monitored application.

Usage: You may use this element to understand the logging requirements at the unit of work level.

Total Log Space Used

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	total_log_used	
Element Type	gauge	
Related Information	• "Unit of Work Log Space Used" on page 162 • "Secondary Logs Allocated Currently" on page 160 • "Maximum Total Log Space Used" on page 159 • "Application with Oldest Transaction" on page 56	

Description: The total amount of active log space currently used (in bytes) in the database.

Usage: Use this element in conjunction with "Total Log Available" on page 163 to determine whether you may need to adjust the following configuration parameters to avoid running out of log space:

- logfilsz
- logprimary
- logsecond

For more information, see the *Administration Guide*.

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Total Log Available

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Resettable	No	
Element Name	total_log_available	
Element Type	water mark	
Related Information	• "Unit of Work Log Space Used" on page 162 • "Secondary Logs Allocated Currently" on page 160 • "Total Log Space Used" on page 162 • "Application with Oldest Transaction" on page 56	

Description: The amount of active log space in the database that is not being used by uncommitted transactions (in bytes).

Usage: Use this element in conjunction with "Total Log Space Used" on page 162 to determine whether you may need to adjust the following configuration parameters to avoid running out of log space:

- logfilsz
- logprimary
- logsecond

For more information, see the *Administration Guide*.

If this value goes down to 0, SQL0964N will be returned. You may need to increase the above configuration parameters, or end the oldest transaction by COMMIT, ROLLBACK or FORCE APPLICATION.

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Database and Application Activity

The following sections provide information on database and application activity.

Locks and Deadlocks

The following elements provide information about locks and deadlocks:

- "Locks Held"
- "Total Lock List Memory In Use" on page 166
- "Deadlocks Detected" on page 166
- "Number of Lock Escalations" on page 167
- "Exclusive Lock Escalations" on page 169
- "Lock Mode" on page 170
- "Lock Status" on page 171
- "Lock Object Type Waited On" on page 172
- "Lock Object Name" on page 173
- "Number of Lock Timeouts" on page 174
- "Maximum Number of Locks Held" on page 174
- "Connections Involved in Deadlock" on page 175
- "Lock Escalation" on page 175
- "Lock Mode Requested" on page 176

Locks Held

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Lock	dbase_lock	Basic
	appl_lock	Basic
Resettable	No	
Element Name	locks_held	
Element Type	gauge	
Related Information	• "Number of Lock Escalations" on page 167 • "Exclusive Lock Escalations" on page 169 • "Maximum Number of Locks Held" on page 174	

Description: The number of locks currently held.

Usage: If the monitor information is at the database level, this is the total number of locks currently held by all applications in the database.

If it is at the application level, this is the total number of locks currently held by all agents for the application. How you use this element depends on the level of information being returned from the database system monitor.

- At the database level, you can use it in one of two ways:
  - This element can provide summary information about locking. For example, you can calculate the average number of locks per application by dividing the value of this element by *Applications Connected Currently*. If the resulting number is high, it may indicate that you can tune one of your applications to improve performance.
  - You can also compare the value of this element against the results of the following formula to determine the number of additional locks that may be requested. This comparison can help you determine if the configuration parameters need adjusting or your applications need tuning.

    ```
    (locklist * 4096 / 36 ) - locks held = # remaining
    ```

    where:
    - *locklist* is the configuration parameter as described in the *Administration Guide*
    - 4096 is the number of bytes in one 4K page
    - 36 is the number of bytes required for each lock.

    Note: You may also use "Total Lock List Memory In Use" on page 166 in a similar fashion.
- At the application level, you can use this counter to determine if the application is approaching the maximum number of locks available to it, as defined by the *maxlocks* configuration parameter. This parameter indicates the percentage of the lock list that each application can use before lock escalations occur. Lock escalations can result in a decrease in concurrency between applications connected to a database. (See the *Administration Guide* for more information about this parameter.)

  Since the *maxlocks* parameter is specified as a percentage and this element is a counter, you can compare the count provided by this element against the total number of locks that can be held by an application, as calculated using the following formula:

  ```
  (locklist * 4096 / 36 ) * (maxlocks / 100)
  ```

  If you have a large number of locks, you may need to perform more commits within your application so that some of the locks can be released.

Total Lock List Memory In Use

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Resettable	No	
Element Name	lock_list_in_use	
Element Type	gauge	
Related Information	• None	

Description: The total amount of lock list memory (in bytes) that is in use.

Usage: This element may be used in conjunction with the *locklist* configuration parameter to calculate the lock list utilization. If the lock list utilization is high, you may want to consider increasing the size of that parameter. See the *Administration Guide* for more information.

Note: When calculating utilization, it is important to note that the *locklist* configuration parameter is allocated in pages of 4K bytes each, while this monitor element provides results in bytes.

Deadlocks Detected

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Lock
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	deadlocks	
Element Type	counter	
Related Information	• “Resetting Monitor Data” on page 25 • “When Counters are Initialized” on page 24 • “Database Activation Timestamp” on page 46 • “Connection Request Start Timestamp” on page 70 • “Number of Lock Escalations” on page 167 • “Exclusive Lock Escalations” on page 169 • “Application ID Holding Lock” on page 182	

Description: The total number of deadlocks that have occurred.

Usage: This element can indicate that applications are experiencing contention problems. These problems could be caused by the following situations:

- Lock escalations are occurring for the database
- An application may be locking tables explicitly when system-generated row locks may be sufficient
- An application may be using an inappropriate isolation level when binding
- Catalog tables are locked for repeatable read
- Applications are getting the same locks in different orders, resulting in deadlock.

You may be able to resolve the problem by determining in which applications (or application processes) the deadlocks are occurring. You may then be able to modify the application to better enable it to execute concurrently. Some applications, however, may not be capable of running concurrently.

You can use the connection timestamp monitor elements (*"Last Reset Timestamp" on page 248*, *"Database Activation Timestamp" on page 46*, and *"Connection Request Start Timestamp" on page 70)* to determine the severity of the deadlocks. For example, 10 deadlocks in 5 minutes is much more severe than 10 deadlocks in 5 hours.

The descriptions for the related elements listed above may also provide additional tuning suggestions.

Number of Lock Escalations

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Transaction	xaction_event	
Element Name	lock_escals	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Database Activation Timestamp" on page 46 • "Exclusive Lock Escalations" on page 169 • "Maximum Number of Locks Held" on page 174	

Description: The number of times that locks have been escalated from several row locks to a table lock.

Usage: A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application, or the lock list space consumed by all applications is approaching the total lock list space. The amount of lock list space available is determined by the *maxlocks* and *locklist* configuration parameters.

When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs.

This data item includes a count of all lock escalations, including exclusive lock escalations.

There are several possible causes for excessive lock escalations:

- The lock list size (*locklist*) may be too small for the number of concurrent applications
- The percent of the lock list usable by each application (*maxlocks*) may be too small
- One or more applications may be using an excessive number of locks.

To resolve these problems, you may be able to:

- Increase the *locklist* configuration parameter value. See the *Administration Guide* for a description of this configuration parameter.
- Increase the *maxlocks* configuration parameter value. See the *Administration Guide* for a description of this configuration parameter.
- Identify the applications with large numbers of locks (see *Maximum Number of Locks Held)*, or those that are holding too much of the lock list, using the following formula:

  ```
  (((locks held * 36) / (locklist * 4096)) * 100)
  ```

  and comparing the value to maxlocks. These applications can also cause lock escalations in other applications by using too large a portion of the lock list. These applications may need to resort to using table locks instead of row locks, although table locks may cause an increase in “Lock Waits” on page 177 and “Time Waited On Locks” on page 178.

Exclusive Lock Escalations

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Transaction	xaction_event	
Element Name	x_lock_escals	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Database Activation Timestamp" on page 46 • "Number of Lock Escalations" on page 167 • "Connection Request Start Timestamp" on page 70 • "Maximum Number of Locks Held" on page 174	

Description: The number of times that locks have been escalated from several row locks to one exclusive table lock, or the number of times an exclusive lock on a row caused the table lock to become an exclusive lock.

Usage: Other applications cannot access data held by an exclusive lock; therefore it is important to track exclusive locks since they can impact the concurrency of your data.

A lock is escalated when the total number of locks held by an application reaches the maximum amount of lock list space available to the application. The amount of lock list space available is determined by the *locklist* and *maxlocks* configuration parameters.

When an application reaches the maximum number of locks allowed and there are no more locks to escalate, it will then use space in the lock list allocated for other applications. When the entire lock list is full, an error occurs.

See *"Number of Lock Escalations" on page 167* for possible causes and resolutions to excessive exclusive lock escalations.

An application may be using exclusive locks when share locks are sufficient. Although share locks may not reduce the total number of lock escalations share lock escalations may be preferable to exclusive lock escalations.

Lock Mode

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	lock	Lock
	lock_wait	Lock
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Element Name	lock_mode	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • Other lock information	

Description: The type of lock being held.

Usage: This mode can help you determine the source of contention for resources.

This element indicates one of the following, depending on the type of monitor information being examined:

- The type of lock another application holds on the object that this application is waiting to lock (for application-monitoring and deadlock-monitoring levels)
- The type of lock held on the object by this application (for object-lock levels).

The values for this field are:

Mode	Type of Lock	API Constant
	No Lock	SQLM_LNON
IS	Intention Share Lock	SQLM_LOIS
IX	Intention Exclusive Lock	SQLM_LOIX
S	Share Lock	SQLM_LOOS
SIX	Share with Intention Exclusive Lock	SQLM_LSIX
X	Exclusive Lock	SQLM_LOOX
IN	Intent None	SQLM_LOIN
Z	Super Exclusive Lock	SQLM_LOOZ
U	Update Lock	SQLM_LOOU
NS	Next Key Share Lock	SQLM_LONS
NX	Next Key Exclusive Lock	SQLM_LONX
W	Weak Exclusive Lock	SQLM_LOOW
NW	Next Key Weak Exclusive Lock	SQLM_LONW

Lock Status

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Resettable	No	
Element Name	lock_status	
Element Type	information	
Related Information	• "Lock Mode" on page 170 • "Lock Object Type Waited On" on page 172 • "Table File ID" on page 196	

Description: Indicates the internal status of the lock.

Usage: This element can help explain what is happening when an application is waiting to obtain a lock on an object. While it may appear that the application already has a lock on the object it needs, it may have to wait to obtain a different type of lock on the same object.

The lock can be in one of the following statuses:

Granted state	indicates that the application has the lock in the state specified by "Lock Mode" on page 170.
Converting state	indicates that the application is trying to change the lock held to a different type; for example, changing from a share lock to an exclusive lock.

Note: API users should refer to the *sqlmon.h* header file containing definitions of database system monitor constants.

Lock Object Type Waited On

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
	appl_lock	Lock
Lock	lock	Basic
	lock_wait	Lock
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Element Name	lock_object_type	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25	

Description: The type of object against which the application holds a lock (for object-lock-level information), or the type of object for which the application is waiting to obtain a lock (for application-level and deadlock-level information).

Usage: This element can help you determine the source of contention for resources.

The objects may be one of the following types:

- Table space
- Table
- Record (or row)
- Internal (another type of lock held internally by the database manager).

Lock Object Name

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	appl_lock	Lock
	lock	Basic

Resettable	No

Event Type	Logical Data Grouping
Deadlock	dlconn_event

Element Name	lock_object_name
Element Type	information

Related Information

- "Resetting Monitor Data" on page 25
- "Lock Object Type Waited On" on page 172
- "Table Space Name" on page 179
- "Table Name" on page 188
- "Table Schema Name" on page 189

Description: This element is provided for informational purposes only. It is the name of the object for which the application holds a lock (for object-lock-level information), or the name of the object for which the application is waiting to obtain a lock (for application-level and deadlock-level information).

Usage: It is the name of the object for table-level locks is the file ID (FID) for SMS and DMS table spaces. For row-level locks, the object name is the row ID (RID). For table space locks, the object name is blank.

To determine the table holding the lock, use *Table Name* and *Table Schema Name* instead of the file ID, since the file ID may not be unique.

To determine the table space holding the lock, use *Table Space Name.*

Lock Node

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Resettable	No
Element Name	lock_node
Element Type	information

Related Information

- None

Description: The node involved in a lock.

Usage: This can be used for troubleshooting.

Number of Lock Timeouts

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	lock_timeouts	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • Other elements in "Locks and Deadlocks" on page 164	

Description: The number of times that a request to lock an object timed-out instead of being granted.

Usage: This element can help you adjust the setting for the *locktimeout* database configuration parameter. If the number of lock time-outs becomes excessive when compared to normal operating levels, you may have an application that is holding locks for long durations. In this case, this element may indicate that you should analyze some of the other elements related to "Locks and Deadlocks" on page 164 to determine if you have an application problem.

You could also have too few lock time-outs if your *locktimeout* database configuration parameter is set too high. In this case, your applications may wait excessively to obtain a lock. See the *Administration Guide* for more information.

Maximum Number of Locks Held

Event Type	**Logical Data Grouping**
Transaction	xaction_event
Element Name	locks_held_top
Element Type	counter
Related Information	• "Locks Held" on page 164 • "Number of Lock Escalations" on page 167 • "Exclusive Lock Escalations" on page 169

Description: The maximum number of locks held during this transaction.

Usage: You can use this element to determine if your application is approaching the maximum number of locks available to it, as defined by the *maxlocks* configuration parameter. This parameter indicates the percentage of the lock list that each application can use before lock escalations occur. Lock escalations can result in a decrease in concurrency between applications connected to a database. (See the *Administration Guide* for more information about this parameter.)

Since the *maxlocks* parameter is specified as a percentage and this element is a counter, you can compare the count provided by this element against the total number of locks that can be held by an application, as calculated using the following formula:

```
(locklist * 4096 / 36 ) * (maxlocks / 100)
```

If you have a large number of locks, you may need to perform more commits within your application so that some of the locks can be released.

Connections Involved in Deadlock

Event Type	**Logical Data Grouping**
Deadlock	deadlock_event
Element Name	dl_conns
Element Type	gauge
Related Information	• None

Description: The number of connections that are involved in the deadlock.

Usage: Use this element in your monitoring application to identify how many deadlock connection event records will follow in the event monitor data stream.

Lock Escalation

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Lock	lock	Lock
	lock_wait	Lock
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Element Name	lock_escalation	
Element Type	information	
Related Information	• other lock data elements	

Description: Indicates whether a lock request was made as part of a lock escalation.

Usage: Use this element to better understand the cause of deadlocks. If you experience a deadlock that involves applications doing lock escalation, you may want to increase the amount of lock memory or change the percentage of locks that any one application can request.

Lock Mode Requested

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Lock	lock_wait	Lock
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Element Name	lock_mode_requested	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • Other lock information	

Description: The lock mode being requested by the application.

Usage: The mode in which the lock was requested by the application. This value can help you determine the source of contention for resources.

Lock Wait Information

The following elements provide information is returned when a DB2 agent working on behalf of an application is waiting to obtain a lock:

- "Lock Waits" on page 177
- "Time Waited On Locks" on page 178
- "Table Space Name" on page 179
- "Current Agents Waiting On Locks" on page 179
- "Total Time Unit of Work Waited on Locks" on page 180
- "Lock Wait Start Timestamp" on page 180
- "Agent ID Holding Lock" on page 181
- "Application ID Holding Lock" on page 182
- "Sequence Number Holding Lock" on page 183
- "Rolled Back Application" on page 183
- "Rolled Back Sequence Number" on page 184

Lock Waits

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Lock
Application	appl	Lock
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	lock_waits	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Connection Request Start Timestamp" on page 70 • "Time Waited On Locks" on page 178	

Description: The total number of times that applications or connections waited for locks.

Usage: At the database level, this is the total number of times that applications have had to wait for locks within this database.

At the application-connection level, this is the total number of times that this connection requested a lock but had to wait because another connection was already holding a lock on the data.

This element may be used with *Time Waited On Locks* to calculate, at the database level, the average wait time for a lock. This calculation can be done at either the database or the application-connection level.

If the average lock wait time is high, you should look for applications that hold many locks, or have lock escalations, with a focus on tuning your applications to improve concurrency, if appropriate. If escalations are the reason for a high average lock wait time, then the values of one or both of the *locklist* and *maxlocks* configuration parameters may be too low. See the *Administration Guide* for more information.

Time Waited On Locks

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Lock
Application	appl	Lock
	appl_lock	

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event
Transaction	xaction_event

Element Name	lock_wait_time
Element Type	counter

Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Current Agents Waiting On Locks" on page 179 • "Lock Waits" on page 177

Description: The total elapsed time waited for a lock.

Usage: At the database level, this is the total amount of elapsed time that all applications were waiting for a lock within this database.

At the application-connection and transaction levels, this is the total amount of elapsed time that this connection or transaction has waited for a lock to be granted to it.

This element may be used in conjunction with the Lock Waits monitor element to calculate the average wait time for a lock. This calculation can be performed at either the database or the application-connection level.

When using data elements providing elapsed times, you should consider:

- Elapsed times are affected by system load, so the more processes you have running, the higher this elapsed time value.
- To calculate this data element at the database level, the database system monitor sums the application-level times. This can result in double counting elapsed times at a database level, since more than one application process can be running at the same time.

  To provide meaningful data, you can calculate the average wait time for a lock, as described above.

Table Space Name

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Buffer Pool
Application	appl_lock	Basic
Lock	lock	Lock
	lock_wait	Lock
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Table Space	tablespace_header	
Element Name	tablespace_name	
Element Type	information	
Related Information	• “Resetting Monitor Data” on page 25 • “Lock Object Type Waited On” on page 172	

Description: The name of a table space.

Usage: This element can help you determine the source of contention for resources.

It is equivalent to the TBSPACE column in the database catalog table SYSCAT.TABLESPACE. At the application level, application-lock level, and deadlock monitoring level, this is the name of the table space that the application is waiting to lock. Another application currently holds a lock on this table space.

At the lock level, this is the name of the table space against which the application currently holds a lock.

At the table space level (when the buffer pool monitor group is ON), this is the name of the table space for which information is returned.

Current Agents Waiting On Locks

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Lock	dbase_lock	Basic
Resettable	No	
Element Name	locks_waiting	
Element Type	gauge	
Related Information	• “Applications Connected Currently” on page 85	

Description: Indicates the number of agents waiting on a lock.

Usage: When used in conjunction with *Applications Connected Currently*, this element indicates the percentage of applications waiting on locks. If this number is high, the applications may have concurrency problems, and you should identify applications that are holding locks or exclusive locks for long periods of time.

Total Time Unit of Work Waited on Locks

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Unit of Work
Resettable	No	
Element Name	uow_lock_wait_time	
Element Type	counter	
Related Information	• “Resetting Monitor Data” on page 25 • Application-level information on locks	

Description: The total amount of elapsed time this unit of work has spent waiting for locks.

Usage: This element can help you determine the severity of the resource contention problem.

Lock Wait Start Timestamp

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Lock
Lock	lock_wait	Lock
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Element Name	lock_wait_start_time	
Element Type	timestamp	
Related Information	• “Resetting Monitor Data” on page 25 • “Agent ID Holding Lock” on page 181	

Description: The date and time that this application started waiting to obtain a lock on the object that is currently locked by another application.

Usage: This element can help you determine the severity of resource contention.

Agent ID Holding Lock

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
	appl_lock	Lock
Lock	lock_wait	Lock

Resettable	No
Element Name	agent_id_holding_lock
Element Type	information
Related Information	• “Resetting Monitor Data” on page 25 • “Lock Wait Start Timestamp” on page 180 • “Application ID Holding Lock” on page 182

Description: The application handle of the agent holding a lock for which this application is waiting. The lock monitor group must be turned on to obtain this information.

Usage: This element can help you determine which applications are in contention for resources.

If this element is 0 (zero) and the application is waiting for a lock, this indicates that the lock is held by an indoubt transaction. You can use either “Application ID Holding Lock” on page 182 or the command line processor LIST INDOUBT TRANSACTIONS command (which displays the application ID of the CICS agent that was processing the transaction when it became indoubt) to determine the indoubt transaction, and then either commit it or roll it back.

Note that more than one application can hold a shared lock on an object for which this application is waiting. See “Lock Mode” on page 170 for information about the type of lock that the application holds. If you are taking an application snapshot, only one of the agent IDs holding a lock on the object will be returned. If you are taking a lock snapshot, all of the agent IDs holding a lock on the object will be identified.

Application ID Holding Lock

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Lock
	appl_lock	Lock
Lock	lock_wait	Lock
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Element Name	appl_id_holding_lk	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • "Agent ID Holding Lock" on page 181 • "Deadlocks Detected" on page 166	

Description: The application ID of the application that is holding a lock on the object that this application is waiting to obtain.

Usage: This element can help you determine which applications are in contention for resources. Specifically, it can help you identify the application handle (agent ID) and table ID that are holding the lock. Note that you may use the LIST APPLICATIONS command to obtain information to relate the application ID with an agent ID. However, it is a good idea to collect this type of information when you take the snapshot, as it could be unavailable if the application ends before you run the LIST APPLICATIONS command.

Note that more than one application can hold a shared lock on an object for which this application is waiting to obtain a lock. See "Lock Mode" on page 170 for information about the type of lock that the application holds. If you are taking an application snapshot, only one of the application IDs holding a lock on the object will be returned. If you are taking a lock snapshot, all of the application IDs holding a lock on the object will be returned.

Sequence Number Holding Lock

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
	appl_lock	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Deadlock	dlconn_event	
Element Name	sequence_no_holding_lk	
Element Type	information	
Related Information	• None	

Description: This element is reserved for future use. In this release, its value will always be “0001”. In future releases of the product, it may contain different values.

Rolled Back Application

Event Type	**Logical Data Grouping**
Deadlock	deadlock_event
Element Name	rolled_back_appl_id
Element Type	information
Related Information	• “Service Level” on page 42 • “Maximum Number of Coordinating Agents” on page 90

Description: Application id that was rolled back when a deadlock occurred.

Usage: A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted

Rolled Back Agent

Event Type	**Logical Data Grouping**
Deadlock	deadlock_event
Element Name	rolled_back_agent_id
Element Type	information
Related Information	• “Service Level” on page 42 • “Maximum Number of Coordinating Agents” on page 90

Description: Agent that was rolled back when a deadlock occurred.

Usage: A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted

Rolled Back Sequence Number

Event Type	**Logical Data Grouping**
Deadlock	deadlock_event
Element Name	rolled_back_sequence_no
Element Type	information
Related Information	• None

Description: The sequence number of the application that was rolled back when a deadlock occurred.

Usage: A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted

Rollforward Monitoring

Recovering database changes can be a time consuming process. You can use the database system monitor to monitor the progression of a recovery. The following elements provide information about rollforward status:

- "Rollforward Timestamp"
- "Tablespace Being Rolled Forward" on page 185
- "Rollforward Type" on page 185
- "Log Being Rolled Forward" on page 185
- "Log Phase" on page 186
- "Number of Rollforward Table Spaces" on page 186

Rollforward Timestamp

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table Space	rollfwd_info	Basic
Resettable	No	
Element Name	rf_timestamp	
Element Type	timestamp	
Related Information	• "Tablespace Being Rolled Forward" on page 185	

Description: The timestamp of the log being processed.

Usage: If a rollforward is in progress, this is the timestamp of the log record being processed. This is an indicator of the data changes that will be recovered.

Tablespace Being Rolled Forward

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table Space	rollfwd_ts_info	Basic
Resettable	No	
Element Name	ts_name	
Element Type	information	
Related Information	• "Rollforward Timestamp" on page 184	

Description: The name of the table space currently rolled forward.

Usage: If a rollforward is in progress, this element identifies the table spaces involved.

Rollforward Type

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table Space	rollfwd_info	Basic
Resettable	No	
Element Name	rf_type	
Element Type	information	
Related Information	• None	

Description: The type of rollforward in progress.

Usage: An indicator of whether recovery is happening at a database or table space level. For more information on rollforward recovery at the database or table space level see the *Administration Guide.*

Log Being Rolled Forward

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table Space	rollfwd_info	Basic
Resettable	No	
Element Name	rf_log_num	
Element Type	information	
Related Information	• None	

Description: The log being processed.

Usage: If a rollforward is in progress, this element identifies the log involved.

Log Phase

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollfwd_info	Basic
Resettable	No	
Element Name	rf_status	
Element Type	information	
Related Information	• None	

Description: The status of the recovery.

Usage: This element indicates the progression of a recovery. It indicates if the recovery is in an undo (rollback) or redo (rollforward) phase.

Number of Rollforward Table Spaces

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollfwd_info	Basic
Resettable	No	
Element Name	rf_num_tspaces	
Element Type	counter	
Related Information	• None	

Description: The number of table spaces involved in a rollforward.

Usage: This is a counter of the table spaces involved in recovery.

Table Activity

The following elements provide information about the tables:

- "Table Type" on page 187
- "Table Name" on page 188
- "Table Schema Name" on page 189
- "Rows Deleted" on page 190
- "Rows Inserted" on page 190
- "Rows Updated" on page 191
- "Rows Selected" on page 191
- "Rows Written" on page 192
- "Rows Read" on page 193

- “Accesses to Overflowed Records” on page 194
- “Internal Rows Deleted” on page 194
- “Internal Rows Updated” on page 195
- “Internal Rows Inserted” on page 196
- “Table File ID” on page 196
- “Page Reorganizations” on page 197

Table Type

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table	table	Table
Resettable	No	
Event Type	**Logical Data Grouping**	
Table	table_event	
Element Name	table_type	
Element Type	information	
Related Information	• “Resetting Monitor Data” on page 25 • “Table File ID” on page 196	

Description: The type of table for which information is returned.

Usage: You can use this element to help identify the table for which information is returned. If the table is a user table or a system catalog table, you can use *Table Name* and *Table Schema Name* to identify the table.

The type of table may be one of the following:

- User table.
- User table that has been dropped. The table type will only be updated after the changes are committed (either explicitly or implicitly).
- Temporary table. Information regarding temporary tables is returned, even though the tables are not kept in the database after being used. You may still find information about this type of table useful.
- System catalog table.
- Reorganization table. A table created and used by the database manager while performing a reorganization of another table.

Table Name

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table
Application	appl	Lock
	sqml_appl_lock	Lock
Lock	lock	Lock
	lock_wait	Lock

Resettable	No
Event Type	**Logical Data Grouping**
Table	table_event
Deadlock	dlconn_event
Element Name	table_name
Element Type	information
Related Information	• "Resetting Monitor Data" on page 25 • "Table Schema Name" on page 189 • "Lock Object Type Waited On" on page 172

Description: The name of the table.

Usage: Along with *Table Schema Name*, this element can help you determine the source of contention for resources.

At the application-level, application-lock level, and deadlock-monitoring-level, this is the table that the application is waiting to lock, because it is currently locked by another application. For snapshot monitoring, this item is only valid when the "lock" monitor group information is turned on, and when *Lock Object Type Waited On* indicates that the application is waiting to obtain a table lock.

For snapshot monitoring at the object-lock level, this item is returned for table-level and row-level locks. The table reported at this level is the table against which this application holds these locks.

For snapshot and event monitoring at the table level, this is the table for which information has been collected. This element is blank for temporary tables, reorganization tables, and tables that were dropped. Table names are only provided for catalog and user tables. For snapshot monitoring, this element is only valid when the "table" monitor group information is turned on.

Table Schema Name

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table
Application	appl	Lock
	appl_lock	Lock
Lock	lock	Lock
	lock_wait	Lock

Resettable	No

Event Type	Logical Data Grouping
Table	table_event
Deadlock	dlconn_event

Element Name	table_schema
Element Type	information
Related Information	• "Resetting Monitor Data" on page 25 • "Table Name" on page 188 • "Lock Object Type Waited On" on page 172

Description: The schema of the table.

Usage: Along with *Table Name,* this element can help you determine the source of contention for resources.

For application-level, application-lock-level, deadlock-monitoring-level, this is the schema of the table that the application is waiting to lock, because it is currently locked by another application. This element is only set if *Lock Object Type Waited On* indicates that the application is waiting to obtain a table lock. For snapshot monitoring at the application-level and application-lock levels, this item is only valid when the "lock" monitor group information is turned on.

For snapshot monitoring at the object-lock level, this item is returned for table and row level locks. The table reported at this level is the table against which this application holds these locks.

For snapshot and event monitoring at the table level, this element identifies the schema of the table for which information has been collected. This element is blank for temporary tables, reorganization tables, and tables that were dropped. Schema names are provided only for catalog and user tables. For snapshot monitoring, this element is valid only when the "table" monitor group information is turned on.

Rows Deleted

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	rows_deleted	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Internal Rows Deleted" on page 194	

Description: This is the number of row deletions attempted.

Usage: You can use this element to gain insight into the current level of activity within the database manager.

This count does not include the attempts counted in *Internal Rows Deleted.*

Rows Inserted

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	rows_inserted	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24	

Description: This is the number of row insertions attempted.

Usage: You can use this element to gain insight into the current level of activity within the database manager.

Rows Updated

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	rows_updated	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Internal Rows Updated" on page 195	

Description: This is the number of row updates attempted.

Usage: You can use this element to gain insight into the current level of activity within the database manager.

This value does not include updates counted in *Internal Rows Updated.* However, rows that are updated by more than one update statement are counted for each update.

Rows Selected

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	rows_selected	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Select SQL Statements Executed" on page 207	

Description: This is the number of rows that have been selected and returned to the application.

Usage: You can use this element to gain insight into the current level of activity within the database manager.

This element does not include a count of rows read for actions such as COUNT(*) or joins.

Rows Written

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table	table	Table
Application	appl	Basic
	stmt	Basic
	subsection	Statement
Dynamic SQL	dynsql	Statement
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Table	table_event	
Statement	stmt_event	
Transaction	xaction_event	
Element Name	rows_written	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "When Counters are Initialized" on page 24 • "Rows Read" on page 193 • "Internal Rows Inserted" on page 196 • "Internal Rows Deleted" on page 194 • "Internal Rows Updated" on page 195	

Description: This is the number of rows changed (inserted, deleted or updated) in the table.

Usage: A high value for table-level information indicates there is heavy usage of the table and you may want to use the Run Statistics (RUNSTATS) utility to maintain efficiency of the packages used for this table.

For application-connections and statements, this element includes the number of rows inserted, updated, and deleted in temporary tables.

At the application, transaction, and statement levels, this element can be useful for analyzing the relative activity levels, and for identifying candidates for tuning.

Rows Read

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table
Application	appl	Basic
	stmt	Basic
	subsection	Statement
Dynamic SQL	dynsql	Statement

Resettable	Yes

Event Type	Logical Data Grouping
Connection	conn_event
Table	table_event
Statement	stmt_event
Transaction	xaction_event

Element Name	rows_read
Element Type	counter

Related Information

- “Resetting Monitor Data” on page 25
- “When Counters are Initialized” on page 24
- “Rows Written” on page 192
- “Accesses to Overflowed Records” on page 194

Description: This is the number of rows read from the table.

Usage: This element helps you identify tables with heavy usage for which you may want to create additional indexes. To avoid the maintenance of unnecessary indexes, you may use the SQL EXPLAIN statement, described in the *Administration Guide* to determine if the package uses an index.

This count is **not** the number of row that were returned to the calling application. Rather, it is the number of rows that had to be read in order to return the result set. For example, the following statement returns one row to the application, but many rows are read to determine the average salary:

```
SELECT AVG(SALARY) FROM USERID.EMPLOYEE
```

This count includes the value in *Accesses to Overflowed Records.*

Accesses to Overflowed Records

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table

Resettable	Yes

Event Type	Logical Data Grouping
Table	table_event

Element Name	overflow_accesses
Element Type	counter
Related Information	• “Resetting Monitor Data” on page 25 • “When Counters are Initialized” on page 24 • “Rows Read” on page 193 • “Rows Written” on page 192

Description: The number of accesses (reads and writes) to overflowed rows of this table.

Usage: Overflowed rows indicate that data fragmentation has occurred. If this number is high, you may be able to improve table performance by reorganizing the table using the REORG utility, which cleans up this fragmentation.

A row overflows if it is updated and no longer fits in the data page where it was originally written. This usually happens as a result of an update of a VARCHAR or an ALTER TABLE statement.

Internal Rows Deleted

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
	stmt	Basic
Dynamic SQL	dynsql	Statement

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event
Statement	stmt_event

Element Name	int_rows_deleted
Element Type	counter
Related Information	• “When Counters are Initialized” on page 24 • “Rows Deleted” on page 190

Description: This is the number of rows deleted from the database as a result of internal activity.

Usage: This element can help you gain insight into internal activity within the database manager of which you might not be aware. If this activity is high, you may want to evaluate your table design to determine if the referential constraints or triggers that you have defined on your database are necessary.

Internal delete activity can be a result of:

- A cascading delete enforcing an ON CASCADE DELETE referential constraint
- A trigger being fired.

Internal Rows Updated

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
	stmt	Basic
Dynamic SQL	dynsql	Statement
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Statement	stmt_event	
Element Name	int_rows_updated	
Element Type	counter	
Related Information	• “When Counters are Initialized” on page 24 • “Rows Updated” on page 191	

Description: This is the number of rows updated from the database as a result of internal activity.

Usage: This element can help you gain insight into internal activity within the database manager of which you might not be aware. If this activity is high, you may want to evaluate your table design to determine if the referential constraints that you have defined on your database are necessary.

Internal update activity can be a result of:

- A *set null* row update enforcing a referential constraint defined with the ON DELETE SET NULL rule
- A trigger being fired.

Internal Rows Inserted

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
	stmt	Basic
Dynamic SQL	dynsql	Statement

Resettable	Yes

Event Type	**Logical Data Grouping**
Database	db_event
Connection	conn_event
Statement	stmt_event

Element Name	int_rows_inserted
Element Type	counter
Related Information	• “When Counters are Initialized” on page 24 • “Rows Inserted” on page 190

Description: The number of rows inserted into the database as a result of internal activity caused by triggers.

Usage: This element can help you gain insight into the internal activity within the database manager. If this activity is high, you may want to evaluate your design to determine if you can alter it to reduce this activity.

Table File ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Lock
Table	table	Table
Lock	appl_lock	Lock
	lock	Lock

Resettable	No
Element Name	table_file_id
Element Type	information
Related Information	• “Resetting Monitor Data” on page 25 • “Table Name” on page 188 • “Table Schema Name” on page 189 • “Table Type” on page 187

Description: This is the file ID (FID) for the table.

Usage: This element is provided for information purposes only. It is returned for compatibility with previous versions of the database system monitor, and it may **not** uniquely identify the table. Use *Table Name* and *Table Schema Name* to identify the table.

Page Reorganizations

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Table	table	Table
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Table	table_event	
Element Name	page_reorgs	
Element Type	counter	
Related Information	• Rows Inserted • Rows Updated	

Description: The number of page reorganizations executed for a table.

Usage: Too many page reorganizations can result in less than optimal insert performance. You can use the REORG TABLE utility to reorganize a table and eliminate fragmentation. You can also use the APPEND parameter for the ALTER TABLE statement to indicate that all inserts are appended at the end of a table and so avoid page reorgs.

In situations where updates to rows causes the row length to increase, the page may have enough space to accommodate the new row, but a page reorg may be required to defragment that space. Or if the page does not have enough space for the new larger row, an overflow record is created being created causing *Accesses to Overflowed Records* during reads. You can avoid both situations by using fixed length columns instead of varying length columns.

SQL Cursors

The following elements provide information about the SQL cursors:
- “Open Remote Cursors” on page 198
- “Open Remote Cursors with Blocking” on page 198
- “Rejected Block Cursor Requests” on page 199
- “Accepted Block Cursor Requests” on page 200
- “Open Local Cursors” on page 201
- “Open Local Cursors with Blocking” on page 201

Open Remote Cursors

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Element Name	open_rem_curs	
Element Type	gauge	
Related Information	• "Open Remote Cursors with Blocking" on page 198 • "Open Local Cursors" on page 201	

Description: The number of remote cursors currently open for this application, including those cursors counted by *Open Remote Cursors with Blocking*.

Usage: You may use this element in conjunction with *Open Remote Cursors with Blocking* to calculate the percentage of remote cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application. See *Open Remote Cursors with Blocking* for more information.

For the number of open cursors used by applications connected to a local database, see *Open Local Cursors*.

Open Remote Cursors with Blocking

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Element Name	open_rem_curs_blk	
Element Type	gauge	
Related Information	• "Open Remote Cursors" on page 198 • "Rejected Block Cursor Requests" on page 199 • "Accepted Block Cursor Requests" on page 200 • "Open Local Cursors" on page 201 • "Open Local Cursors with Blocking" on page 201	

Description: The number of remote blocking cursors currently open for this application.

Usage: You can use this element in conjunction with *Open Remote Cursors* to calculate the percentage of remote cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application:

- Check the pre-compile options for record blocking for treatment of ambiguous cursors
- Redefine cursors to allow for blocking (for example, if possible, specify FOR FETCH ONLY on your cursors).

Rejected Block Cursor Requests and *Accepted Block Cursor Requests* provide additional information that may help you tune your configuration parameters to improve row blocking in your application.

For the number of open blocking cursors used by applications connected to a local database see *Open Local Cursors with Blocking.*

Rejected Block Cursor Requests

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Element Name	rej_curs_blk	
Element Type	counter	
Related Information	• “Accepted Block Cursor Requests” on page 200 • “Open Local Cursors” on page 201 • “Open Local Cursors with Blocking” on page 201 • “Open Remote Cursors” on page 198 • “Open Remote Cursors with Blocking” on page 198	

Description: The number of times that a request for an I/O block at server was rejected and the request was converted to non-blocked I/O.

Usage: If there are many cursors blocking data, the communication heap may become full. When this heap is full, an error is not returned. Instead, no more I/O blocks are allocated for blocking cursors. If cursors are unable to block data, performance can be affected.

If a large number of cursors were unable to perform data blocking, you may be able to improve performance by:

- Increasing the size of the *query_heap* database manager configuration parameter. For more information see the *Administration Guide*.

Accepted Block Cursor Requests

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Element Name	acc_curs_blk	
Element Type	counter	
Related Information	• “Rejected Block Cursor Requests” on page 199 • “Open Local Cursors” on page 201 • “Open Local Cursors with Blocking” on page 201 • “Open Remote Cursors” on page 198 • “Open Remote Cursors with Blocking” on page 198	

Description: The number of times that a request for an I/O block was accepted.

Usage: You can use this element in conjunction with *Rejected Block Cursor Requests* to calculate the percentage of blocking requests that are accepted and/or rejected.

See *Rejected Block Cursor Requests* for suggestions on how to use this information to tune your configuration parameters.

Open Local Cursors

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Element Name	open_loc_curs	
Element Type	gauge	
Related Information	• "Open Local Cursors with Blocking" on page 201 • "Open Remote Cursors" on page 198 • "Open Remote Cursors with Blocking" on page 198 • "Rejected Block Cursor Requests" on page 199 • "Accepted Block Cursor Requests" on page 200	

Description: The number of local cursors currently open for this application, including those cursors counted by *Open Local Cursors with Blocking.*

Usage: You may use this element in conjunction with *Open Local Cursors with Blocking* to calculate the percentage of local cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application.

For cursors used by remote applications, see *Open Remote Cursors.*

Open Local Cursors with Blocking

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	No	
Element Name	open_loc_curs_blk	
Element Type	gauge	
Related Information	• "Open Local Cursors" on page 201 • "Open Remote Cursors" on page 198 • "Open Remote Cursors with Blocking" on page 198 • "Rejected Block Cursor Requests" on page 199 • "Accepted Block Cursor Requests" on page 200	

Description: The number of local blocking cursors currently open for this application.

Usage: You may use this element in conjunction with *Open Local Cursors* to calculate the percentage of local cursors that are blocking cursors. If the percentage is low, you may be able to improve performance by improving the row blocking in the application:

- Check the pre-compile options for record blocking for treatment of ambiguous cursors
- Redefine cursors to allow for blocking (for example, if possible, specify FOR FETCH ONLY on your cursors).

Rejected Block Cursor Requests and Accepted Block Cursor Requests provide additional information that may help you tune your configuration parameters to improve row blocking in your application.

For blocking cursors used by remote applications, see *Open Remote Cursors with Blocking.*

SQL Statement Activity

The following elements provide information about SQL statement activity:

- "Static SQL Statements Attempted" on page 203
- "Dynamic SQL Statements Attempted" on page 203
- "Failed Statement Operations" on page 204
- "Commit Statements Attempted" on page 205
- "Rollback Statements Attempted" on page 206
- "Select SQL Statements Executed" on page 207
- "Update/Insert/Delete SQL Statements Executed" on page 208
- "Data Definition Language (DDL) SQL Statements" on page 208
- "Internal Automatic Rebinds" on page 209
- "Internal Commits" on page 210
- "Internal Rollbacks" on page 211
- "Internal Rollbacks Due To Deadlock" on page 213
- "SQL Requests Since Last Commit" on page 213
- "Statement Node" on page 214
- "Binds/Precompiles Attempted" on page 214

Static SQL Statements Attempted

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	static_sql_stmts	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Failed Statement Operations" on page 204	

Description: The number of static SQL statements that were attempted.

Usage: You can use this element to calculate the total number of successful SQL statements at the database or application level:

```
  Dynamic SQL Statements Attempted
+ Static SQL Statements Attempted
- Failed Statement Operations
= throughput during monitoring period
```

Dynamic SQL Statements Attempted

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	dynamic_sql_stmts	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Failed Statement Operations" on page 204	

Description: The number of dynamic SQL statements that were attempted.

Usage: You can use this element to calculate the total number of successful SQL statements at the database or application level:

```
  Dynamic SQL Statements Attempted
+ Static SQL Statements Attempted
- Failed Statement Operations
= throughput during monitoring period
```

Failed Statement Operations

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	failed_sql_stmts	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Dynamic SQL Statements Attempted" on page 203 • "Static SQL Statements Attempted" on page 203	

Description: The number of SQL statements that were attempted, but failed.

Usage: You can use this element to calculate the total number of successful SQL statements at the database or application level:

```
  Dynamic SQL Statements Attempted
+ Static SQL Statements Attempted
- Failed Statement Operations
= throughput during monitoring period
```

This count includes all SQL statements that received a negative SQLCODE.

This element may also help you in determining reasons for poor performance, since failed statements mean time wasted by the database manager and as a result, lower throughput for the database.

Commit Statements Attempted

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event

Element Name	commit_sql_stmts
Element Type	counter

Related Information
- "When Counters are Initialized" on page 24
- "Internal Commits" on page 210
- "Rollback Statements Attempted" on page 206
- "Internal Rollbacks" on page 211
- "Internal Rollbacks Due To Deadlock" on page 213

Description: The total number of SQL COMMIT statements that have been attempted.

Usage: A small rate of change in this counter during the monitor period may indicate that applications are not doing frequent commits, which may lead to problems with logging and data concurrency.

You can also use this element to calculate the total number of units of work by calculating the sum of the following:

```
   commit statements attempted
 + internal commits
 + rollback statements attempted
 + internal rollbacks
```

Note: The units of work calculated will only include those since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

This calculation can be done at a database or application level.

Rollback Statements Attempted

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	rollback_sql_stmts	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Statement Type" on page 216 • "Commit Statements Attempted" on page 205 • "Internal Commits" on page 210 • "Internal Rollbacks" on page 211 • "Internal Rollbacks Due To Deadlock" on page 213	

Description: The total number of SQL ROLLBACK statements that have been attempted.

Usage: A rollback can result from an application request, a deadlock, or an error situation. This element **only** counts the number of rollback statements issued from applications.

At the application level, this element can help you determine the level of database activity for the application and the amount of conflict with other applications. At the database level, it can help you determine the amount of activity in the database and the amount of conflict between applications on the database.

Note: You should try to minimize the number of rollbacks, since higher rollback activity results in lower throughput for the database.

It may also be used to calculate the total number of units of work, by calculating the sum of the following:

```
  commit statements attempted
+ internal commits
+ rollback statements attempted
+ internal rollbacks
```

Select SQL Statements Executed

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Table Space	tablespace	Basic
Application	appl	Basic

Resettable	Yes

Event Type	Logical Data Grouping
Database	db_event
Connection	conn_event

Element Name	select_sql_stmts
Element Type	counter
Related Information	• "When Counters are Initialized" on page 24 • "Static SQL Statements Attempted" on page 203 • "Dynamic SQL Statements Attempted" on page 203

Description: The number of SQL SELECT statements that were executed.

Usage: You can use this element to determine the level of database activity at the application or database level.

You can also use the following formula to determine the ratio of SELECT statements to the total statements:

```
      select SQL statements executed
    / ( static SQL statements attempted
      + dynamic SQL statements attempted )
```

This information can be useful for analyzing application activity and throughput.

Update/Insert/Delete SQL Statements Executed

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	uid_sql_stmts	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Static SQL Statements Attempted" on page 203 • "Dynamic SQL Statements Attempted" on page 203	

Description: The number of SQL UPDATE, INSERT, and DELETE statements that were executed.

Usage: You can use this element to determine the level of database activity at the application or database level.

You can also use the following formula to determine the ratio of UPDATE, INSERT and DELETE statements to the total number of statements:

```
   update/insert/delete SQL statements executed
 / (static SQL statements attempted + dynamic SQL statements attempted )
```

This information can be useful for analyzing application activity and throughput.

Data Definition Language (DDL) SQL Statements

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	ddl_sql_stmts	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24	

Description: This element indicates the number of SQL Data Definition Language (DDL) statements that were executed.

Usage: You can use this element to determine the level of database activity at the application or database level. DDL statements are expensive to run due to their impact on the system catalog tables. As a result, if the value of this element is high, you should determine the cause, and possibly restrict this activity from being performed.

You can also use this element to determine the percentage of DDL activity using the following formula:

```
 data definition language (DDL) SQL statements / total number of statements
```

This information can be useful for analyzing application activity and throughput. DDL statements can also impact the package cache, by invalidating sections that are stored there and causing additional system overhead due to section recompilation.

Examples of DDL statements are CREATE TABLE, CREATE VIEW, ALTER TABLE, and DROP INDEX.

Internal Automatic Rebinds

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	int_auto_rebinds	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Binds/Precompiles Attempted" on page 214	

Description: The number of automatic rebinds (or recompiles) that have been attempted.

Usage: Automatic rebinds are the internal binds the system performs when an package has been invalidated. The rebind is performed the first time that the database manager needs to execute an SQL statement from the package. For example, packages are invalidated when you:

- Drop an object, such as a table, view, or index, on which the plan is dependent
- Add or drop a foreign key
- Revoke object privileges on which the plan is dependent.

You can use this element to determine the level of database activity at the application or database level. Since internal automatic rebinds can have a significant impact on performance, they should be minimized where possible.

You can also use this element to determine the percentage of rebind activity using the following formula:

```
 internal automatic rebinds / total number of statements
```

This information can be useful for analyzing application activity and throughput.

Internal Commits

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	int_commits	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Commit Statements Attempted" on page 205 • "Rollback Statements Attempted" on page 206 • "Internal Rollbacks" on page 211	

Description: The total number of commits initiated internally by the database manager.

Usage: An internal commit may occur during any of the following:

- A reorganization
- An import
- A bind or pre-compile
- An application ends without executing an explicit SQL COMMIT statement (on UNIX).

This value, which does not include explicit SQL COMMIT statements, represents the number of these internal commits since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

You can use this element to calculate the total number of units of work by calculating the sum of the following:

```
   commit statements attempted
 + internal commits
 + rollback statements attempted
 + internal rollbacks
```

Note: The units of work calculated will only include those since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

This calculation can be done at the application or the database level.

Internal Rollbacks

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	int_rollbacks	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Commit Statements Attempted" on page 205 • "Internal Commits" on page 210 • "Rollback Statements Attempted" on page 206 • "Internal Rollbacks Due To Deadlock" on page 213	

Description: The total number of rollbacks initiated internally by the database manager.

Usage: An internal rollback occurs when any of the following **cannot** complete successfully:

- A reorganization
- An import
- A bind or pre-compile
- An application ends as a result of a deadlock situation or lock timeout situation
- An application ends without executing an explicit commit or rollback statement (on Windows).

This value represents the number of these internal rollbacks since the later of:
- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

While this value does not include explicit SQL ROLLBACK statements, the count from Internal Rollbacks Due To Deadlock is included.

You can use this element to calculate the total number of units of work by calculating the sum of the following:

```
   commit statements attempted
 + internal commits
 + rollback statements attempted
 + internal rollbacks
```

Note: The units of work calculated will include those since the later of:
- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

This calculation can be done at the application or the database level.

Internal Rollbacks Due To Deadlock

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Connection	conn_event	
Element Name	int_deadlock_rollbacks	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24 • "Deadlocks Detected" on page 166 • "Rollback Statements Attempted" on page 206 • "Internal Rollbacks" on page 211	

Description: The total number of forced rollbacks initiated by the database manager due to a deadlock. A rollback is performed on the current unit of work in an application selected by the database manager to resolve the deadlock.

Usage: This element shows the number of deadlocks that have been broken and can be used as an indicator of concurrency problems. It is important, since internal rollbacks due to deadlocks lower the throughput of the database.

This value is included in the value given by Internal Rollbacks.

SQL Requests Since Last Commit

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic
Resettable	No	
Element Name	sql_reqs_since_commit	
Element Type	information	
Related Information	• None	

Description: Number of SQL requests that have been submitted since the last commit.

Usage: You can use this element to monitor the progress of a transaction.

Note: This element is similar to the *cur_reqs* field in the *sqlestat* output. See "Appendix D. DB2 Version 1 sqlestat Users" on page 393 for more information on sqlestat equivalent data elements.

Statement Node

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	stmt	Statement
Resettable	No	
Element Name	stmt_node_number	
Element Type	information	
Related Information	• None	

Description: Node where the statement was executed.

Usage: Used to correlate each statement with the node where it was executed.

Binds/Precompiles Attempted

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Basic
Application	appl	Basic
Resettable	Yes	
Event Type	**Logical Data Grouping**	
Database	db_event	
Connection	conn_event	
Element Name	binds_precompiles	
Element Type	counter	
Related Information	• "When Counters are Initialized" on page 24	
	• "Internal Automatic Rebinds" on page 209	

Description: The number of binds and pre-compiles attempted.

Usage: You can use this element to gain insight into the current level of activity within the database manager.

This value does not include the count of *Internal Automatic Rebinds,* but it does include binds that occur as a result of the REBIND PACKAGE command.

SQL Statement Details

The following elements provide details about the SQL statements:

- “Statement Type” on page 216
- “Statement Operation” on page 217
- “Package Name” on page 218
- “Section Number” on page 219
- “Cursor Name” on page 220
- “Application Creator” on page 220
- “Statement Operation Start Timestamp” on page 221
- “Statement Operation Stop Timestamp” on page 221
- “Event Stop Time” on page 222
- “Event Start Time” on page 222
- “Most Recent Statement Elapsed Time” on page 223
- “SQL Dynamic Statement Text” on page 223
- “Statement Sorts” on page 224
- “Number of Successful Fetches” on page 225
- “SQL Communications Area (SQLCA)” on page 225
- “Query Number of Rows Estimate” on page 226
- “Query Cost Estimate” on page 227

Statement Type

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Statement
	stmt	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	stmt_event	
Element Name	stmt_type	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • "SQL Dynamic Statement Text" on page 223 • "Application Creator" on page 220 • "Section Number" on page 219 • "Package Name" on page 218	

Description: The type of statement processed.

Usage: You can use this element to determine the type of statement that is executing. It can be one of the following:

- A static SQL statement
- A dynamic SQL statement
- An operation other than an SQL statement; for example, a bind or pre-compile operation.

For the snapshot monitor, this element describes the statement that is currently being processed or was most recently processed.

Note: API users should refer to the *sqlmon.h* header file containing definitions of database system monitor constants.

Statement Operation

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic
	stmt	Statement
DCS Application	dcs_appl	Basic
DCS Statement	dcs_stmt	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	stmt_event	
Element Name	stmt_operation (Snapshot) operation (Event)	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • "Statement Type" on page 216 • "SQL Dynamic Statement Text" on page 223 • "Application Creator" on page 220 • "Section Number" on page 219 • "Package Name" on page 218 • "Number of Successful Fetches" on page 225	

Description: The statement operation currently being processed or most recently processed (if none currently running).

Usage: You can use this element to determine the operation that is executing or recently finished.

It can be one of the following.

For SQL operations:

- SELECT
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- OPEN
- FETCH
- CLOSE
- DESCRIBE
- STATIC COMMIT
- STATIC ROLLBACK

- FREE LOCATOR
- PREP_COMMIT
- CALL
- PREP_OPEN
- PREP_EXEC
- COMPILE

For non-SQL operations:
- RUN STATISTICS
- REORG
- REBIND
- REDISTRIBUTE
- GET TABLE AUTHORIZATION
- GET ADMINISTRATIVE AUTHORIZATION

Note: API users should refer to the *sqlmon.h* header file containing definitions of database system monitor constants.

Package Name

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Statement
	stmt	Statement
DCS Application	dcs_appl	Statement
DCS Statement	dcs_stnt	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	stmt_event	
Element Name	package_name	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • "Application Creator" on page 220 • "Section Number" on page 219 • "SQL Dynamic Statement Text" on page 223	

Description: The name of the package that contains the SQL statement currently executing.

Usage: You may use this element to help identify the application program and the SQL statement that is executing.

Section Number

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Statement
	stmt	Statement
DCS Application	dcs_appl	Statement
DCS Statement	dcs_stmt	Statement

Resettable	No

Event Type	**Logical Data Grouping**
Statement	stmt_event

Element Name	section_number
Element Type	information

Related Information	• "Resetting Monitor Data" on page 25 • "SQL Dynamic Statement Text" on page 223 • "Application Creator" on page 220 • "Package Name" on page 218

Description: The internal section number in the package for the SQL statement currently processing or most recently processed.

Usage: For static SQL, you can use this element along with Application Creator and Package Name to query the SYSCAT.STATEMENTS system catalog table and obtain the static SQL statement text, using the sample query as follows:

```
SELECT SEQNO, SUBSTR(TEXT,1,120)
       FROM SYSCAT.STATEMENTS
       WHERE PKGNAME   = 'package_name' AND
             PKGSCHEMA = 'creator'      AND
             SECTNO    = section_number
       ORDER BY SEQNO
```

Note: Exercise caution in obtaining static statement text, because this query against the system catalog table could cause lock contentions. Whenever possible, only use this query when there is little other activity against the database.

Cursor Name

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement
	stmt	Statement

Resettable	No

Event Type	Logical Data Grouping
Statement	stmt_event

Element Name	cursor_name
Element Type	information

Related Information

- “Resetting Monitor Data” on page 25
- “SQL Dynamic Statement Text” on page 223
- “Statement Type” on page 216
- “Number of Successful Fetches” on page 225

Description: The name of the cursor corresponding to this SQL statement.

Usage: You may use this element to identify the SQL statement that is processing. This name will be used on an OPEN, FETCH, CLOSE, and PREPARE of an SQL SELECT statement. If a cursor is not used, this field will be blank.

Application Creator

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement
	stmt	Statement
DCS Application	dcs_appl	Statement
DCS Statement	dcs_stmt	Statement

Resettable	No

Event Type	Logical Data Grouping
Statement	stmt_event

Element Name	creator
Element Type	information

Related Information

- “Resetting Monitor Data” on page 25
- “Package Name” on page 218
- “Section Number” on page 219

Description: The authorization ID of the user that pre-compiled the application.

Usage: You may use this element to help identify the SQL statement that is processing, in conjunction with the CREATOR column of the package section information in the catalogs.

Statement Operation Start Timestamp

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement
	stmt	Statement
DCS Application	dcs_appl	Statement
DCS Statement	dcs_stmt	Statement

Resettable	No
Element Name	stmt_start
Element Type	timestamp
Related Information	• "Resetting Monitor Data" on page 25 • "Statement Operation Stop Timestamp" on page 221 • "Statement Operation" on page 217

Description: The date and time when the Statement Operation started executing.

Usage: You can use this element with Statement Operation Stop Timestamp to calculate the elapsed statement operation execution time.

Statement Operation Stop Timestamp

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Application	dcs_stmt	Statement

Resettable	No
Element Name	stmt_stop
Element Type	Timestamp
Related Information	• "Resetting Monitor Data" on page 25 • "Statement Operation Start Timestamp" on page 221 • "Statement Operation" on page 217 • "Event Stop Time" on page 222

Description: The date and time when the Statement Operation stopped executing.

Usage: You can use this element with Statement Operation Start Timestamp to calculate the elapsed statement operation execution time.

Event Stop Time

Event Type	**Logical Data Grouping**
Statement	stmt_event
Element Name	stop_time
Element Type	timestamp
Related Information	• "Previous Transaction Stop Time" on page 76 • "Statement Operation Stop Timestamp" on page 221

Description: The date and time when the statement stopped executing.

Usage: You can use this element with *Event Start Time* to calculate the elapsed statement execution time.

For a FETCH statement event, this is the time of the last successful fetch.

Event Start Time

Event Type	**Logical Data Grouping**
Database	evmon_start_event
Transaction	xaction_event
Statement	stmt_event
Deadlock	deadlock_event dlconn_event
Element Name	start_time
Element Type	timestamp
Related Information	• "Previous Transaction Stop Time" on page 76 • "Statement Operation" on page 217

Description: The date and time of unit of work start, statement start, or deadlock detection.

This element, in the evmon_start_event API structure indicates the start of the event monitor.

Usage: You can use this element to correlate the deadlock connection records to the deadlock event record, and in conjunction with *Event Stop Time* to calculate the elapsed statement or transaction execution time.

Most Recent Statement Elapsed Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Statement	stmt	Statement
DCS Statement	dcs_stmt	Statement
Resettable	No	
Element Name	stmt_elapsed_time	
Element Type	time	
Related Information	• "Communication Errors" on page 266 • "Communication Error Time" on page 267	

Description: The elapsed execution time of the most recently completed statement.

Usage: Use this element as an indicator of the time it takes for a statement to complete.

SQL Dynamic Statement Text

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	stmt	Statement
Dynamic SQL	dynsql	Basic
DCS Statement	dcs_stmt	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	stmt_event	
Element Name	stmt_text	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • "Statement Operation" on page 217 • "Cursor Name" on page 220 • "Input Database Alias" on page 248 • "Application Creator" on page 220 • "Package Name" on page 218 • "Section Number" on page 219	

Description: This is the text of the dynamic SQL statement.

Usage: For application snapshots, this statement text helps you identify what the application was executing when the snapshot was taken, or most recently processed if no statement was being processed right at the time the snapshot was taken.

For dynamic SQL statements, this element identifies the SQL text associated with a package.

For event monitors, it is returned in the Statement event record for all dynamic statements.

See Section Number for information on how to query the system catalog tables to obtain static SQL statement text that is not provided due to performance considerations.

Statement Sorts

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Statement
	stmt	Statement
Dynamic SQL	dynsql	Statement
Resettable	No	
Element Name	stmt_sorts	
Element Type	counter	
Related Information	• "Resetting Monitor Data" on page 25 • "Total Sorts" on page 98	

Description: The total number of times that a set of data was sorted in order to process the statement operation.

Usage: You can use this element to help identify the need for an index, since indexes can reduce the need for sorting of data. Using the related elements in the above table you can identify the SQL statement for which this element is providing sort information, and then analyze this statement to determine index candidates by looking at columns that are being sorted (for example, columns used in ORDER BY and GROUP BY clauses and join columns). See **explain** in the *Administration Guide* for information on checking whether your indexes are used to optimize sort performance.

This count includes sorts of temporary tables that were generated internally by the database manager to execute the statement. The number of sorts is associated with the first FETCH operation of the SQL statement. This information is returned to you when the operation for the statement is the first FETCH. You should note that for blocked cursors several fetches may be performed when the cursor is opened. In these cases it can be difficult to use the snapshot monitor to obtain the number of sorts, since a snapshot would need to be taken while DB2 was internally issuing the first FETCH.

A more reliable way to determine the number of sorts performed when using a blocked cursor would be with an event monitor declared for statements. The total sorts counter, in the statement event for the CLOSE cursor, contains the total number of sorts that were performed while executing the statement for which the cursor was defined.

Number of Successful Fetches

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	stmt_event	
Element Name	fetch_count	
Element Type	counter	
Related Information	• "Statement Type" on page 216 • "Statement Operation" on page 217 • "Cursor Name" on page 220 • "Statement Operation Start Timestamp" on page 221 • "Statement Operation Stop Timestamp" on page 221	

Description: The number of successful fetches performed on a specific cursor.

Usage: You can use this element to gain insight into the current level of activity within the database manager.

For performance reasons, a statement event monitor does not generated a statement event record for every FETCH statement. A record event is only generated when a FETCH returns a non-zero SQLCODE.

SQL Communications Area (SQLCA)

Event Type	**Logical Data Grouping**
Statement	stmt_event
Element Name	sqlca
Element Type	information
Related Information	• "Statement Operation" on page 217

Description: The SQLCA data structure that was returned to the application at statement completion.

Usage: The SQLCA data structure can be used to determined if the statement completed successfully. See the *SQL Reference* or *Administrative API Reference* for information about the content of the SQLCA.

Query Number of Rows Estimate

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement
Resettable	No	
Element Name	query_card_estimate	
Element Type	information	
Related Information	• "Resetting Monitor Data" on page 25 • "Query Cost Estimate" on page 227	

Description: An estimate of the number of rows that will be returned by a query.

Usage: This estimate by the SQL compiler can be compared with the run time actuals.

This data element also returns information for the following SQL statements when you are monitoring DB2 Connect.

- INSERT, UPDATE, and DELETE

  Indicates the number of rows affected.
- PREPARE

  Estimate of the number of rows that will be returned. Only collected if the DRDA server is DB2 Universal Database, DB2 for VM and VSE, or DB2 for OS/400.
- FETCH

  Set to the number of rows fetched. Only collected if the DRDA server is DB2 for OS/400.

If information is not collected for a DRDA server, then the data element is set to zero.

Query Cost Estimate

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement
Resettable	No	
Element Name	query_cost_estimate	
Element Type	information	
Related Information	• None	

Description: Estimated cost, in timerons, for a query, as determined by the SQL compiler.

Usage: This allows correlation of actual run-time with the compile-time estimates.

This data element also returns information for the following SQL statements when you are monitoring DB2 Connect.

- PREPARE

  Represents the relative cost of the prepared SQL statement.
- FETCH

  Contains the length of the row retrieved. Only collected if the DRDA server is DB2 for OS/400.

If information is not collected for a DRDA server, then the data element is set to zero.

Note: If the DRDA server is DB2 for OS/390, this estimate could be higher than 2**32 - 1 (the maximum integer number that can be expressed through an unsigned long variable). In that case, the value returned by the System Monitor for this data element will be 2**32 - 1.

Subsection Details

When a statement is executed against a partitioned database, it is divided into subsections that may be executed on different nodes. An application may have several subsections simultaneously executing on a node. See “Monitoring Subsections” on page 30 and the *Administration Guide* for more information on subsections.

For problem determination, you may have to locate the problem subsection. For example, a subsection may be waiting on a tablequeue, because one of the writers to this tablequeue is in lock wait on another node. To get the overall picture for an application, you may have to issue an application snapshot on each node where the application is running.

The following database system monitor elements provide information about Subsections:

- "Subsection Number"
- "Subsection Node Number" on page 229
- "Subsection Status" on page 229
- "Execution Elapsed Time" on page 230
- "Number of Agents Working on a Subsection" on page 230
- "Waiting for Any Node to Send on a Tablequeue" on page 230
- "Waited for Node on a Tablequeue" on page 231
- "Total Number of Tablequeue Buffers Overflowed" on page 231
- "Current Number of Tablequeue Buffers Overflowed" on page 232
- "Number of Rows Read from Tablequeues" on page 233
- "Number of Rows Written to Tablequeues" on page 233

Subsection Number

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	ss_number	
Element Type	information	
Related Information	• None	

Description: Identifies the subsection associated with the returned information.

Usage: This number relates to the subsection number in the access plan that can be obtained with db2expln (see *Administration Guide*).

Subsection Node Number

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	ss_node_number	
Element Type	information	
Related Information	• None	

Description: Node where the subsection was executed.

Usage: Use to correlate each subsection with the database partition where it was executed.

Subsection Status

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Resettable	No	
Element Name	ss_status	
Element Type	information	
Related Information	• "Waited for Node on a Tablequeue" on page 231 • "Waiting for Any Node to Send on a Tablequeue" on page 230	

Description: The current status of an executing subsection.

Usage: The current status values can be:

- executing
- waiting for a lock
- waiting to receive data on a tablequeue
- waiting to send data on a tablequeue

Execution Elapsed Time

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	ss_exec_time	
Element Type	counter	
Related Information	• None	

Description: The time in seconds that it took a subsection to execute.

Usage: Allows you to track the progress of a subsection.

Number of Agents Working on a Subsection

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	num_subagents	
Element Type	gauge	
Related Information	• None	

Description: Total number of subagents currently working on a subsection.

Usage: Indicates the current degree of parallelism. Helps you track how execution is progressing.

Waiting for Any Node to Send on a Tablequeue

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Resettable	No	
Element Name	tq_wait_for_any	
Element Type	information	
Related Information	• “Subsection Status” on page 229 • “Waited for Node on a Tablequeue” on page 231	

Description: This flag is used to indicate that the subsection is blocked because it is waiting to receive rows from any node.

Usage: If Subsection Status indicates *waiting to receive data on a tablequeue* and this flag is TRUE, then the subsection is waiting to receive rows from any node. This generally indicates that the SQL statement has not processed to the point it can pass data to the waiting agent. For example, the writing agent may be performing a sort and will not write rows until the sort has completed. From the db2expln output, determine the subsection number associated with the tablequeue that the agent is waiting to receive rows from. You can then examine the status of that subsection by taking a snapshot on each node where it is executing.

Waited for Node on a Tablequeue

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	subsection	Statement
Resettable	No	
Element Name	tq_node_waited_for	
Element Type	information	
Related Information	• "Subsection Status" on page 229 • "Waiting for Any Node to Send on a Tablequeue" on page 230	

Description: If the subsection status Subsection Status is *waiting to receive* or *waiting to send* and Waiting for Any Node to Send on a Tablequeue is FALSE, then this is the number of the node that this agent is waiting for.

Usage: This can be used for troubleshooting. You may want to take an application snapshot on the node that the subsection is waiting for. For example, the application could be in a lock wait on that node.

Total Number of Tablequeue Buffers Overflowed

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	tq_tot_send_spills	
Element Type	counter	
Related Information	• "Subsection Status" on page 229 • "Current Number of Tablequeue Buffers Overflowed" on page 232	

Description: Total number of tablequeue buffers overflowed to a temporary table.

Usage: Indicates the total number of tablequeue buffers that have been written to a temporary table. See “Current Number of Tablequeue Buffers Overflowed” for more information.

Current Number of Tablequeue Buffers Overflowed

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	subsection	Statement
Resettable	No	
Element Name	tq_cur_send_spills	
Element Type	gauge	
Related Information	• “Subsection Status” on page 229 • “Total Number of Tablequeue Buffers Overflowed” on page 231	

Description: Current number of tablequeue buffers residing in a temporary table.

Usage: An agent writing to a tablequeue may be sending rows to several readers. The writing agent will overflow buffers to a temporary table when the agent that it is currently sending rows to is not accepting rows and another agent requires rows in order to proceed. Overflowing to temporary table allows both the writer and the other readers to continue processing.

Rows that have been overflowed will be sent to the reading agent when it is ready to accept more rows.

If this number is high, and queries fail with sqlcode -968, and there are messages in *db2diad.log* indicating that your ran out of temporary space in the TEMP table space, then tablequeue overflows may be the cause. This could indicate a problem on another node (such as locking). You would investigate by taking snapshots on all the partitions for this query.

There are also cases, perhaps because of the way data is partitioned, where many buffers need to be overflowed for the query. In these cases you will need to add more disk to the temporary table space.

Number of Rows Read from Tablequeues

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	tq_rows_read	
Element Type	counter	
Related Information	• None	

Description: Total number of rows read from tablequeues.

Usage: If monitoring does not indicate that this number is increasing, then processing progress is not taking place.

If there is significant differences in this number between nodes, then some nodes may be over utilized while others are being under utilized.

If this number is large, then there is a lot of data being shipped between nodes, suggest that optimization might improve the access plan.

Number of Rows Written to Tablequeues

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	tq_rows_written	
Element Type	counter	
Related Information	• None	

Description: Total number of rows written to tablequeues.

Usage: If monitoring does not indicate that this number is increasing, then processing progress is not taking place.

If there is significant differences in this number between nodes, then some nodes may be over utilized while others are being under utilized.

If this number is large, then there is a lot of data being shipped between nodes, suggest that optimization might improve the access plan.

Maximum Number of Tablequeue Buffers Overflows

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	tq_max_send_spills	
Element Type	water mark	
Related Information	• "Total Number of Tablequeue Buffers Overflowed" on page 231 • "Current Number of Tablequeue Buffers Overflowed" on page 232	

Description: Maximum number of tablequeue buffers overflowed to a temporary table.

Usage: Indicates the maximum number of tablequeue buffers that have been written to a temporary table.

Waited on Node on a Tablequeue

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Resettable	No	
Element Name	tq_id_waiting_on	
Element Type	information	
Related Information	• "Subsection Status" on page 229 • "Waited for Node on a Tablequeue" on page 231	

Description: The agent that is waiting.

Usage: This can be used for troubleshooting.

Dynamic SQL

The DB2 statement cache stores packages and statistics for frequently used SQL statements. By examining the contents of this cache, you can identify the dynamic SQL statements that are most frequently executed, and the queries that consume the most resource. Using this information, you can examine the most commonly executed and most expensive SQL operations, to determine if SQL tuning could result in better database performance.

- "Statement Executions" on page 235

- "Statement Compilations"
- "Statement Worst Preparation Time" on page 236
- "Statement Best Preparation Time" on page 236
- "Elapsed Statement Execution Time" on page 236

Statement Executions

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Dynamic SQL	dynsql	Basic
Resettable	Yes	
Element Name	num_executions	
Element Type	counter	
Related Information	• "Statement Compilations" on page 235	

Description: The number of times that an SQL statement has been executed.

Usage: You can use this element to identify the most frequently executed SQL statements in your system.

Statement Compilations

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Dynamic SQL	dynsql	Basic
Resettable	Yes	
Element Name	num_compilations	
Element Type	counter	
Related Information	• "Statement Executions" on page 235	

Description: The number of different compilations for a specific SQL statement.

Usage: Some SQL statements issued on different schemas, such as ″select t1 from foo″ will appear to be the same statement in the DB2 cache even though they refer to different access plans. Use this value in conjunction with Statement Executions to determine whether a bad compilation environment may be skewing the results of dynamic SQL snapshot statistics.

Statement Worst Preparation Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Dynamic SQL	dynsql	Basic
Resettable	No	
Element Name	prep_time_worst	
Element Type	water mark	
Related Information	• “Statement Best Preparation Time” on page 236	

Description: The longest amount of time in microseconds that was required to prepare a specific SQL statement.

Usage: Use this value in conjunction with Statement Best Preparation Time to identify SQL statements that are expensive to compile.

Statement Best Preparation Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Dynamic SQL	dynsql	Basic
Resettable	No	
Element Name	prep_time_best	
Element Type	water mark	
Related Information	• “Statement Worst Preparation Time” on page 236	

Description: The shortest amount of time that was required to prepare a specific SQL statement.

Usage: Use this value in conjunction with Statement Worst Preparation Time to identify SQL statements that are expensive to compile.

Elapsed Statement Execution Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Dynamic SQL	dynsql	Statement
Resettable	Yes	
Element Name	total_exec_time	
Element Type	time	
Related Information	• “Statement Executions” on page 235 • “Statement Compilations” on page 235 • “Total System CPU for a Statement” on page 246 • “Total User CPU for a Statement” on page 247	

Description: The total time in seconds and microseconds that was spent executing a particular statement in the SQL cache.

Usage: Use this element with “Statement Executions” on page 235 determine the average elapsed time for the statement and identify the SQL statements that would most benefit from a tuning of their SQL. The “Statement Compilations” on page 235 must be considered when evaluating the contents of this data element.

Intra-query Parallelism

The following database system monitor elements provide information about queries for which the degree of parallelism is greater than 1:

- “Number of Agents Working on a Statement”
- “Number of Agents Created” on page 238
- “Degree of Parallelism” on page 238

Number of Agents Working on a Statement

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Statement	stmt	Statement
	subsection	Statement
Resettable	No	
Element Name	num_agents	
Element Type	gauge	
Related Information	• “Number of Agents Created” on page 238 • “Degree of Parallelism” on page 238	

Description: Number of concurrent agents currently executing a statement or subsection.

Usage: An indicator how well the query is parallelized. This is useful for tracking the progress of query execution, by taking successive snapshots.

Number of Agents Created

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database	dbase	Statement
Application	stmt	Statement
Resettable	No	
Element Name	agents_top	
Element Type	water mark	
Related Information	• "Number of Agents Working on a Statement" on page 237 • "Degree of Parallelism" on page 238	

Description: At the application level, this is the maximum number of agents that were used when executing the statement. At the database level, it is the maximum number of agents for all applications.

Usage: An indicator how well intra-query parallelism was realized.

Degree of Parallelism

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Statement	stmt	Statement
Resettable	No	
Element Name	degree_parallelism	
Element Type	information	
Related Information	• "Number of Agents Working on a Statement" on page 237 • "Number of Agents Created" on page 238	

Description: The degree of parallelism requested when the query was bound.

Usage: Use with "Number of Agents Created", to determine if the query achieved maximum level of parallelism.

CPU Usage

The CPU usage for an application is broken down into **user CPU**, which is the CPU consumed while executing application code, and **system** CPU, which is the CPU consumed executing system calls.

CPU consumption is available at the application, transaction, statement, and subsection levels.

- “User CPU Time used by Agent”
- “System CPU Time used by Agent” on page 240
- “User CPU Time used by Statement” on page 241
- “System CPU Time used by Statement” on page 242
- “User CPU Time” on page 243
- “System CPU Time” on page 244
- “User CPU Time used by Subsection” on page 245
- “System CPU Time used by Subsection” on page 246
- “Total System CPU for a Statement” on page 246
- “Total User CPU for a Statement” on page 247

User CPU Time used by Agent

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	Yes	
Element Name	agent_usr_cpu_time	
Element Type	time	
Related Elements	• “System CPU Time used by Agent” on page 240 • “User CPU Time used by Statement” on page 241 • “System CPU Time used by Statement” on page 242 • “User CPU Time used by Subsection” on page 245 • “System CPU Time used by Subsection” on page 246 • “User CPU Time” on page 243 • “System CPU Time” on page 244 • “Total System CPU for a Statement” on page 246 • “Total User CPU for a Statement” on page 247	

Description: The total CPU time (in seconds and microseconds) used by database manager agent process.

Usage: This element along with the other CPU-time related elements can help you identify applications or queries that consume large amounts of CPU.

This counter includes time spent on both SQL and non-SQL statements, as well as any fenced user defined functions (UDF) or stored procedures executed by the application.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

Note: If this information is not available for your operating system, this element will be returned as 0. For example, they are not available on OS/2.

System CPU Time used by Agent

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Basic
Resettable	Yes	
Element Name	agent_sys_cpu_time	
Element Type	time	
Related Information	• "User CPU Time used by Agent" on page 239 • "User CPU Time used by Statement" on page 241 • "System CPU Time used by Statement" on page 242 • "User CPU Time used by Subsection" on page 245 • "System CPU Time used by Subsection" on page 246 • "User CPU Time" on page 243 • "System CPU Time" on page 244 • "Total System CPU for a Statement" on page 246 • "Total User CPU for a Statement" on page 247	

Description: The total *system* CPU time (in seconds and microseconds) used by the database manager agent process.

Usage: This element along with the other related CPU-time elements can help you understand the level of activity within an application, and may help you identify applications that could benefit from additional tuning.

It includes CPU time for both SQL and non-SQL statements, as well as CPU time for any fenced User Defined Functions (UDFs)

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

Note: If this information is not available for your operating system, this element will be set to 0. For example, it is not available for OS/2.

User CPU Time used by Statement

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl	Statement
	stmt	Statement
Resettable	No	
Element Name	stmt_usr_cpu_time	
Element Type	time	
Related Information	• "System CPU Time used by Agent" on page 240 • "User CPU Time used by Agent" on page 239 • "System CPU Time used by Statement" on page 242 • "User CPU Time used by Subsection" on page 245 • "System CPU Time used by Subsection" on page 246 • "User CPU Time" on page 243 • "System CPU Time" on page 244 • "Total System CPU for a Statement" on page 246 • "Total User CPU for a Statement" on page 247	

Description: The total *user* CPU time (in seconds and microseconds) used by the currently executing statement.

Usage: This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

This counter includes time spent on both SQL and non-SQL statements, as well as any fenced user defined functions (UDF) or stored procedures executed by the application.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

Note: If this information is not available for your operating system, this element will be set to 0. For example, it is not available for OS/2.

System CPU Time used by Statement

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement
	stmt	Statement
Resettable	No	
Element Name	stmt_sys_cpu_time	
Element Type	time	
Related Information	• "System CPU Time used by Agent" on page 240 • "User CPU Time used by Statement" on page 241 • "User CPU Time used by Agent" on page 239 • "User CPU Time used by Subsection" on page 245 • "System CPU Time used by Subsection" on page 246 • "User CPU Time" on page 243 • "System CPU Time" on page 244 • "Total System CPU for a Statement" on page 246 • "Total User CPU for a Statement" on page 247	

Description: The total *system* CPU time (in seconds and microseconds) used by the currently executing statement.

Usage: This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

This counter includes time spent on both SQL and non-SQL statements, as well as any fenced user defined functions (UDF) or stored procedures executed by the application.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

Note: If this information is not available for your operating system, this element will be set to 0. For example, it is not available for OS/2.

User CPU Time

Event Type	**Logical Data Grouping**
Connection	conn_event
Transaction	xaction_event
Statement	stmt_event
Element Name	user_cpu_time
Element Type	time
Related Information	• "System CPU Time used by Agent" on page 240 • "User CPU Time used by Statement" on page 241 • "System CPU Time used by Statement" on page 242 • "User CPU Time used by Subsection" on page 245 • "System CPU Time used by Subsection" on page 246 • "User CPU Time used by Agent" on page 239 • "System CPU Time" on page 244 • "Total System CPU for a Statement" on page 246 • "Total User CPU for a Statement" on page 247

Description: The total *user* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement.

Usage: This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

Note: If this information is not available for your operating system, this element will be set to 0. For example, it is not available for OS/2.

System CPU Time

Event Type	**Logical Data Grouping**
Connection	conn_event
Transaction	xaction_event
Statement	stmt_event

Element Name	system_cpu_time
Element Type	time
Related Information	• “System CPU Time used by Agent” on page 240 • “User CPU Time used by Statement” on page 241 • “System CPU Time used by Statement” on page 242 • “User CPU Time used by Subsection” on page 245 • “System CPU Time used by Subsection” on page 246 • “User CPU Time” on page 243 • “User CPU Time used by Agent” on page 239 • “Total System CPU for a Statement” on page 246 • “Total User CPU for a Statement” on page 247

Description: The total *system* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement.

Usage: This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications help could benefit from additional tuning.

Note: If this information is not available for your operating system, this element will be set to 0. For example, it is not available for OS/2.

User CPU Time used by Subsection

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	subsection	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	ss_usr_cpu_time	
Element Type	time	
Related Information	• "System CPU Time used by Agent" on page 240 • "User CPU Time used by Statement" on page 241 • "System CPU Time used by Statement" on page 242 • "User CPU Time used by Agent" on page 239 • "System CPU Time used by Subsection" on page 246 • "User CPU Time" on page 243 • "System CPU Time" on page 244 • "Total System CPU for a Statement" on page 246 • "Total User CPU for a Statement" on page 247	

Description: The total user CPU time (in seconds and microseconds) used by the currently executing statement subsection.

Usage: This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

System CPU Time used by Subsection

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Statement	subsection_event	
Element Name	ss_sys_cpu_time	
Element Type	time	
Related Information	• "System CPU Time used by Agent" on page 240 • "User CPU Time used by Statement" on page 241 • "System CPU Time used by Statement" on page 242 • "User CPU Time used by Subsection" on page 245 • "System CPU Time used by Subsection" on page 246 • "User CPU Time" on page 243 • "User CPU Time used by Agent" on page 239 • "Total System CPU for a Statement" on page 246 • "Total User CPU for a Statement" on page 247	

Description: The total system CPU time (in seconds and microseconds) used by the currently executing statement subsection.

Usage: This element along with the other related CPU-time elements can help you understand the level of activity within an application, and can help you identify applications that could benefit from additional tuning.

System CPU represents the time spent in system calls. User CPU represents time spent executing database manager code.

Total System CPU for a Statement

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Statement
Resettable	Yes	
Element Name	tot_s_cpu_time	
Element Type	time	
Related Information	• "Total User CPU for a Statement" on page 247 • "Elapsed Statement Execution Time" on page 236	

Description: The total system CPU time for an SQL statement.

Usage: Use this element with Elapsed Statement Execution Time and Total User CPU for a Statement to evaluate which statements are the most expensive.

Total User CPU for a Statement

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Dynamic SQL	dynsql	Statement
Resettable	Yes	
Element Name	tot_u_cpu_time	
Element Type	time	
Related Information	• "Total System CPU for a Statement" on page 246 • "Elapsed Statement Execution Time" on page 236	

Description: The total user CPU time for an SQL statement.

Usage: Use this element with Elapsed Statement Execution Time and to evaluate the longest running statements.

Snapshot Monitoring Elements

The following elements provide information about monitoring applications. They are returned as output for every snapshot:

- "Last Reset Timestamp" on page 248
- "Input Database Alias" on page 248
- "Snapshot Time" on page 249
- "Number of Nodes in Partition" on page 249

Last Reset Timestamp

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Database	dbase	Basic
Application	appl	Basic
Table Space	tablespace_header	Buffer Pool
Table	table_header	Table
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic

Resettable	No
Element Name	last_reset
Element Type	timestamp
Related Information	• “Resetting Monitor Data” on page 25 • “Input Database Alias” on page 248

Description: Indicates the date and time that the monitor counters were reset for the application issuing the GET SNAPSHOT.

Usage: You can use this element to help you determine the scope of information returned by the database system monitor.

If the counters have never been reset, this element will be zero.

The database manager counters will only be reset if you reset all active databases.

Input Database Alias

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl_id_info	Basic
Table Space	tablespace_header	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool
Table	table_header	Table
Lock	dbase_lock	Basic

Resettable	No
Element Name	input_db_alias
Element Type	information
Related Information	• “Resetting Monitor Data” on page 25 • “Last Reset Timestamp” on page 248 • “Database Alias Used by Application” on page 61

Description: The alias of the database provided when calling the snapshot function.

Usage: This element can be used to identify the specific database to which the monitor data applies. It contains blanks unless you requested monitor information related to a specific database.

The value of this field may be different than the value of the *Database Alias Used by Application* monitor element since a database can have many different aliases. Different applications and users can use different aliases to connect to the same database.

Snapshot Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	collected	Basic
Resettable	No	
Element Name	time_stamp	
Element Type	timestamp	
Related Information	• None	

Description: The date and time when the database system monitor information was collected.

Usage: You can use this element to help relate data chronologically if you are saving the results in a file or database for ongoing analysis.

Number of Nodes in Partition

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
Resettable	No	
Event Type	**Logical Data Grouping**	
Database Manager	log_header_event	
Element Name	num_nodes_in_db2_instance	
Element Type	information	
Related Information	• None	

Description: The number of nodes on the instance where the snapshot was taken.

Usage: Use this element to determine the number of nodes for an instance. For non-partitioned system databases, this value will be 1.

Event Monitoring Elements

The following elements provide information about monitoring applications. They are returned as output for events:

- "Number of Event Monitor Overflows"
- "Time of First Event Overflow"
- "Time of Last Event Overflow" on page 251
- "Byte Order of Event Data" on page 251
- "Version of Monitor Data" on page 252
- "Event Monitor Name" on page 252
- "Partial Record" on page 253
- "Event Time" on page 253

Number of Event Monitor Overflows

Event Type	**Logical Data Grouping**
Overflow Record	overflow_event
Element Name	count
Element Type	counter
Related Information	• None

Description: The number of consecutive overflows that have occurred.

Usage: You may use this element to get an indication of how much monitor data has been lost.

The event monitor sends one overflow record for a set of consecutive overflows.

Time of First Event Overflow

Event Type	**Logical Data Grouping**
Overflow Record	overflow_event
Element Name	first_overflow_time
Element Type	timestamp
Related Information	• "Number of Event Monitor Overflows" on page 250

Description: The date and time of the first overflow recorded by this overflow record.

Usage: Use this element with *Time of Last Event Overflow* to calculate the elapsed time for which the overflow record was generated.

Time of Last Event Overflow

Event Type	**Logical Data Grouping**
Overflow Record	overflow_event
Element Name	last_overflow_time
Element Type	timestamp
Related Information	• "Number of Event Monitor Overflows" on page 250

Description: The date and time of the last overflow recorded this overflow record.

Usage: Use this element with *Time of First Event Overflow* to calculate the elapsed time for which the overflow record was generated.

Byte Order of Event Data

Event Type	**Logical Data Grouping**
Event Log Header	log_header_event
Element Name	byte_order
Element Type	information
Related Information	• None

Description: The byte ordering of numeric data, specifically whether the event data stream was generated on a "big endian" server (for example, a RISC System/6000) or "little endian" server (for example, a PS/2).

Usage: This information is needed to allow you to interpret numeric data in the data stream, since the byte order of integers on a "big endian" server is the reverse of the byte order on a "little endian" server.

If the application that processes the data recognizes that it is running on one type of computer hardware (for example, a big endian computer), while the event data was produced on the other type of computer hardware (for example, a little endian computer), then the monitoring application will have to reverse the bytes of numeric data fields before interpreting them. Otherwise, byte reordering is not required.

This element can be set to one of the following API constants:
- SQLM_BIG_ENDIAN
- SQLM_LITTLE_ENDIAN

Version of Monitor Data

Event Type	Logical Data Grouping
Event Log Header	log_header_event
Element Name	version
Element Type	information
Related Information	• None

Description: The version of the database manager that produced the event monitor data stream.

Usage: The data structures used by the event monitor may change between releases of the database manager. As a result, your monitor applications should check the version of the data stream to determine if they can process the data they will be receiving.

For this release, this element is set to the API constant SQLM_DBMON_VERSION6.

Event Monitor Name

Event Type	Logical Data Grouping
Event Log Header	log_header_event
Element Name	event_monitor_name
Element Type	information
Related Information	• None

Description: The name of the event monitor that created the event data stream.

Usage: This element allows you to correlate the data that you are analyzing to a specific event monitor in the system catalog tables. This is the same name that can be found in the NAME column of the SYSCAT.EVENTMONITORS catalog table, which is the name specified on the CREATE EVENT MONITOR and SET EVENT MONITOR statements.

Partial Record

Event Type	**Logical Data Grouping**
Database	db_event
Table	table_event
Table Space	tablespace_event
	bufferpool_event
Connection	conn_event
Statement	stmt_event
	subsection_event
Transaction	xaction_event

Element Name	partial_record
Element Type	information
Related Information	• None

Description: Indicates that an event monitor record is only a partial record.

Usage: Most event monitors do not output their results until database deactivation. You can use the FLUSH EVENT MONITORS statement to force monitor values to the event monitor output writer (see "FLUSH EVENT MONITOR" on page 323). This allows you to force event monitor records to the writer without needing to stop and restart the event monitor. This data element indicates whether an event monitor record was the result of flush operation and so is a partial record.

Flushing an event monitor does not cause its values to be reset. This means that a complete event monitor record is still generated when the event monitor is triggered.

Event Time

Event Type	**Logical Data Grouping**
Table Space	tablespace_event
Table	table_event

Element Name	event_time
Element Type	information
Related Information	• None

Description: The date and time an event occurred.

Usage: You can use this element to help relate events chronologically.

DB2 Connect

The following elements provide DB2 Connection information at the database, application, transaction, and statement levels:

- "DCS Database Name" on page 255
- "Host Database Name" on page 255
- "Database Alias at the Gateway" on page 256
- "DB2 Connect Gateway First Connect Initiated" on page 256
- "Maximum Number of Concurrent Connections" on page 256
- "Total Number of Attempted Connections for DB2 Connect" on page 257
- "Current Number of Connections for DB2 Connect" on page 257
- "Number of Connections Waiting for the Host to Reply" on page 258
- "Number of Connections Waiting for the Client to Send Request" on page 258
- "Elapsed Time Spent on DB2 Connect Gateway Processing" on page 259
- "Number of SQL Statements Attempted" on page 259
- "Number of Open Cursors" on page 259
- "DCS Application Status" on page 260
- "Host Coded Character Set ID" on page 261
- "Outbound Communication Protocol" on page 261
- "Outbound Communication Address" on page 262
- "Inbound Communication Address" on page 262
- "Inbound Number of Bytes Received" on page 263
- "Outbound Number of Bytes Sent" on page 263
- "Outbound Number of Bytes Received" on page 264
- "Inbound Number of Bytes Sent" on page 264
- "Transaction ID" on page 265
- "Host Response Time" on page 265
- "Most Recent Response Time for Connect" on page 266
- "Most Recent Connection Elapsed Time" on page 266
- "Communication Errors" on page 266
- "Communication Error Time" on page 267

DCS Database Name

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl_info	Basic

Resettable	No
Element Name	dcs_db_name
Element Type	information
Related Information	• "Host Database Name" on page 255 • "Database Alias at the Gateway" on page 256

Description: The name of the DCS database as catalogued in the DCS directory.

Usage: Use this element for problem determination on DCS applications.

Host Database Name

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl_info	Basic

Resettable	No
Element Name	host_db_name
Element Type	information
Related Information	• "DCS Database Name" on page 255 • "Database Alias at the Gateway" on page 256

Description: The real name of the host database for which information is being collected or to which the application is connected. This is the name that was given to the database when it was created.

Usage: Use this element for problem determination on DCS applications.

Database Alias at the Gateway

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	gw_db_alias	
Element Type	information	
Related Information	• "DCS Database Name" on page 255 • "Host Database Name" on page 255	

Description: The alias used at the DB2 Connect gateway to connect to the host database.

Usage: Use this element for problem determination on DCS applications.

DB2 Connect Gateway First Connect Initiated

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
Resettable	No	
Element Name	gw_con_time	
Element Type	timestamp	
Related Information	• None	

Description: The date and time when the first connection to the host database was initiated from the DB2 Connect gateway.

Usage: Use this element for problem determination on DCS applications.

Maximum Number of Concurrent Connections

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
Resettable	No	
Element Name	gw_connections_top	
Element Type	water mark	
Related Information	• "Total Number of Attempted Connections for DB2 Connect" on page 257 • "Current Number of Connections for DB2 Connect" on page 257	

Description: The maximum number of concurrent connections to a host database that have been handled by the DB2 Connect gateway since the first database connection.

Usage: This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

Total Number of Attempted Connections for DB2 Connect

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic
Resettable	Yes	
Element Name	gw_total_cons	
Element Type	water mark	
Related Information	• "Maximum Number of Concurrent Connections" on page 256 • "Current Number of Connections for DB2 Connect" on page 257	

Description: The total number of connections attempted from the DB2 Connect gateway since the last `db2start` command or the last reset.

Usage: This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

Current Number of Connections for DB2 Connect

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic
Resettable	No	
Element Name	gw_cur_cons	
Element Type	gauge	
Related Information	• "Maximum Number of Concurrent Connections" on page 256 • "Total Number of Attempted Connections for DB2 Connect" on page 257	

Description: The current number of connections to host databases being handled by the DB2 Connect gateway.

Usage: This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

Number of Connections Waiting for the Host to Reply

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic
Resettable	No	
Element Name	gw_cons_wait_host	
Element Type	gauge	
Related Information	• "Current Number of Connections for DB2 Connect" on page 257 • "Number of Connections Waiting for the Client to Send Request" on page 258	

Description: The current number of connections to host databases being handled by the DB2 Connect gateway that are waiting for a reply from the host.

Usage: This value can change frequently. It should be sampled at regular intervals over an extended period in order to obtain a realistic view of gateway usage.

Number of Connections Waiting for the Client to Send Request

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic
Resettable	No	
Element Name	gw_cons_wait_client	
Element Type	gauge	
Related Information	• "Current Number of Connections for DB2 Connect" on page 257 • "Number of Connections Waiting for the Host to Reply" on page 258	

Description: The current number of connections to host databases being handled by the DB2 Connect gateway that are waiting for the client to send a request.

Usage: This value can change frequently. It should be sampled at regular intervals over an extended period in order to obtain a realistic view of gateway usage.

Elapsed Time Spent on DB2 Connect Gateway Processing

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Application	dcs_appl	Statement
DCS Statement	dcs_stmt	Statement
Resettable	Yes (at application) No (at other levels)	
Element Name	gw_exec_time	
Element Type	time	
Related Information	• None	

Description: The time in seconds and microseconds at the DB2 Connect gateway to process an application request (since the connection was established), or to process a single statement.

Usage: Use this element to determine what portion of the overall processing time is due to DB2 Connect gateway processing.

Number of SQL Statements Attempted

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
Resettable	Yes	
Element Name	sql_stmts	
Element Type	counter	
Related Information	• Snapshot Time	

Description: The number of SQL statements that have been attempted since the latter of: application start up, database activation, or last reset.

Usage: Use this element to measure database activity at the database or application level. To calculate the SQL statement throughput for a given period, you can divide this element by the elapsed time between two snapshots.

Number of Open Cursors

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Application	dcs_appl	Basic
Resettable	No	
Element Name	open_cursors	
Element Type	gauge	
Related Information	• None	

Description: The number of cursors currently open for an application.

Usage: Use this element to assess how much memory is being allocated. The amount of memory allocated by the DB2 client, DB2 Connect, or the database agent on the target database is related to the number of cursors that are currently open. Knowing this information can help with capacity planning. For example, each open cursor that is doing blocking has a buffer size of `RQRIOBLK`. If *deferred_prepare* is enabled, then two buffers will be allocated.

DCS Application Status

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	dcs_appl_status	
Element Type	information	
Related Information	• "Host Coded Character Set ID" on page 261 • "Outbound Communication Protocol" on page 261 • "Outbound Communication Address" on page 262 • "Inbound Communication Address" on page 262	

Description: The status of a DCS application at the DB2 Connect gateway.

Usage: Use this element for problem determination on DCS applications. Values are:

- SQLM_DCS_CONNECTPEND_OUTBOUND

  The application has initiated a database connection from the DB2 Connect gateway to the host database, but the request has not completed yet.
- SQLM_DCS_UOWWAIT_OUTBOUND

  The DB2 Connect gateway is waiting for the host database to reply to the application's request.
- SQLM_DCS_UOWWAIT_INBOUND

  The connection from the DB2 Connect gateway to the host database has been established and the gateway is waiting for SQL requests from the application. Or the DB2 Connect gateway is waiting on behalf of the unit of work in the application. This usually means that the application's code is being executed.

Host Coded Character Set ID

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	host_ccsid	
Element Type	information	
Related Information	• "DCS Application Status" on page 260 • "Outbound Communication Protocol" on page 261 • "Outbound Communication Address" on page 262 • "Inbound Communication Address" on page 262	

Description: This is the coded character set identifier (CCSID) of the host database.

Usage: Use this element for problem determination on DCS applications.

Outbound Communication Protocol

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	outbound_comm_protocol	
Element Type	information	
Related Information	• "DCS Application Status" on page 260 • "Host Coded Character Set ID" on page 261 • "Outbound Communication Address" on page 262 • "Inbound Communication Address" on page 262	

Description: The communication protocol used between the DB2 Connect gateway and the host.

Usage: Use this element for problem determination on DCS applications. Valid values are:

- SQLM_PROT_APPC
- SQLM_PROT_TCPIP

Outbound Communication Address

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_info	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	outbound_comm_address	
Element Type	information	
Related Information	• "DCS Application Status" on page 260 • "Host Coded Character Set ID" on page 261 • "Outbound Communication Protocol" on page 261 • "Inbound Communication Address" on page 262	

Description: This is the communication address of the target database. For example, it could be an SNA net ID and LU partner name, or an IP address and port number for TCP/IP.

Usage: Use this element for problem determination on DCS applications.

Inbound Communication Address

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
Application	appl_info	Basic
DCS Application	dcs_appl_info	Basic
Resettable	No	
Element Name	inbound_comm_address	
Element Type	information	
Related Information	• "DCS Application Status" on page 260 • "Host Coded Character Set ID" on page 261 • "Outbound Communication Protocol" on page 261 • "Outbound Communication Address" on page 262	

Description: This is the communication address of the client. For example, it could be an SNA net ID and LU partner name, or an IP address and port number for TCP/IP.

Usage: Use this element for problem determination on DCS applications.

Inbound Number of Bytes Received

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Basic
DCS Statement	dcs_stmt	Statement
Resettable	Yes (at application) No (at other levels)	
Element Name	inbound_bytes_received	
Element Type	counter	
Related Information	• "Outbound Number of Bytes Sent" on page 263 • "Outbound Number of Bytes Received" on page 264 • "Inbound Number of Bytes Sent" on page 264	

Description: The number of bytes received by the DB2 Connect gateway from the client, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

Usage: Use this element to measure the throughput from the client to the DB2 Connect gateway.

Outbound Number of Bytes Sent

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
DCS Statement	dcs_stmt	Statement
Resettable	No (at statement) Yes (at other levels)	
Element Name	outbound_bytes_sent	
Element Type	counter	
Related Information	• "Inbound Number of Bytes Received" on page 263 • "Outbound Number of Bytes Received" on page 264 • "Inbound Number of Bytes Sent" on page 264	

Description: The number of bytes sent by the DB2 Connect gateway to the host, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

Usage: Use this element to measure the throughput from the DB2 Connect gateway to the host database.

Outbound Number of Bytes Received

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
DCS Statement	dcs_stmt	Statement
Resettable	No (at statement) Yes (at other levels)	
Element Name **Element Type**	outbound_bytes_received counter	
Related Information	• "Inbound Number of Bytes Received" on page 263 • "Outbound Number of Bytes Sent" on page 263 • "Inbound Number of Bytes Sent" on page 264	

Description: The number of bytes received by the DB2 Connect gateway from the host, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

Usage: Use this element to measure the throughput from the host databases to the DB2 Connect gateway.

Inbound Number of Bytes Sent

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Basic
DCS Statement	dcs_stmt	Statement
Resettable	Yes (at application) No (at other levels)	
Element Name **Element Type**	inbound_bytes_sent counter	
Related Information	• "Inbound Number of Bytes Received" on page 263 • "Outbound Number of Bytes Sent" on page 263 • "Outbound Number of Bytes Received" on page 264	

Description: The number of bytes sent by the DB2 Connect gateway to the client, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

Usage: Use this element to measure the throughput from the DB2 Connect gateway to the client.

Transaction ID

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Application	dcs_appl	Unit of Work
Resettable	No	
Element Name	xid	
Element Type	information	
Related Information	• None	

Description: A unique transaction identifier (across all databases) generated by a transaction manager in a two-phase commit transaction.

Usage: This identifier can be used to correlate the transaction generated by the transaction manager with the transactions executed against multiple databases. It can be used to help diagnose transaction manager problems by tying database transactions involving a two-phase commit protocol with the transactions originated by the transaction manager.

Host Response Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
DCS Statement	dcs_stmt	Statement
Resettable	No (at statement) Yes (at other levels)	
Element Name	host_response_time	
Element Type	time	
Related Information	• "Outbound Number of Bytes Received" on page 264 • "Outbound Number of Bytes Sent" on page 263	

Description: At the DCS statement level, this is the elapsed time between the time that the statement was sent from the DB2 Connect gateway to the host for processing and the time when the result was received from the host. At other levels, it is the sum of the elapsed times for all the statements that were executed for a particular application or database.

Usage: Use this element with Outbound Number of Bytes Sent and Outbound Number of Bytes Received to calculate the outbound response time (transfer rate):

```
(outbound bytes sent + outbound bytes received) / host response time
```

Most Recent Response Time for Connect

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Database	dcs_dbase	Basic
Resettable	No	
Element Name	con_response_time	
Element Type	time	
Related Information	• "Package Cache Overflows" on page 153	

Description: The elapsed time between the start of connection processing and actual establishment of a connection, for the most recent DCS application that connected to this database.

Usage: Use this element as an indicator of the time it currently takes applications to connect to a particular host database.

Most Recent Connection Elapsed Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Database	dcs_dbase	Basic
Resettable	No	
Element Name	con_elapsed_time	
Element Type	time	
Related Information	• "Package Cache Overflows" on page 153	

Description: The elapsed time that the DCS application that most recently disconnected from this host database was connected.

Usage: Use this element as an indicator of the length of time that applications are maintaining connections to a host database.

Communication Errors

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Database	dcs_dbase	Basic
Resettable	Yes	
Element Name	gw_comm_errors	
Element Type	counter	
Related Information	• "Communication Error Time" on page 267 • "Most Recent Statement Elapsed Time" on page 223	

Description: The number of times that a communication error (SQL30081) occurred while a DCS application was attempting to connect to a host database, or while it was processing an SQL statement.

Usage: By monitoring the number of communication errors over time, you can assess whether your DB2 Connect gateway has connectivity problems with a particular host database. You can establish what you consider to be a normal error threshold, so that any time the number of errors exceeds this threshold an investigation of the communication errors should be made.

Use this element for problem determination, in conjunction with the communication error logged in `db2diag.log`.

Communication Error Time

Snapshot Level	**Logical Data Grouping**	**Monitor Switch**
DCS Database	dcs_dbase	Basic
Resettable	No	
Element Name	gw_comm_error_time	
Element Type	timestamp	
Related Information	• “Communication Errors” on page 266	

Description: The date and time when the most recent communication error (SQL30081) occurred while a DCS application was attempting to connect to a host database, or while it was processing an SQL statement.

Usage: Use this element for problem determination, in conjunction with Communication Error and the communication error logged in `db2diag.log`.

Transaction Processor Monitoring

In a transaction monitor or application server (multi-tier) environment, application users do not issue SQL requests directly. Instead, they request the transaction processor monitor (for example, CICS, TUXEDO, or ENCINA running on a UNIX, OS/2, or Windows NT server) or application server to execute a business transaction. Each business transaction is an application part that issues SQL requests to the database server. Because the SQL requests are issued by an intermediate server, the database server has no information about the original client that caused the execution of the SQL request.

Developers of transaction processor monitor (TP monitor) transactions or application server code can use the sqleseti - Set Client Information API to provide information about the original client to the database server. This information can be found in the following data elements:

- “TP Monitor Client User ID” on page 268

- “TP Monitor Client Workstation Name” on page 268
- “TP Monitor Client Application Name” on page 269
- “TP Monitor Client Accounting String” on page 269.

TP Monitor Client User ID

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Basic
Resettable	No	
Element Name	tpmon_client_userid	
Element Type	information	
Related Information	• “TP Monitor Client Workstation Name” on page 268 • “TP Monitor Client Application Name” on page 269 • “TP Monitor Client Accounting String” on page 269	

Description: The client user ID generated by a transaction manager and provided to the server, if the **sqleseti** API is used.

Usage: Use this element in application server or TP monitor environments to identify the end-user for whom the transaction is being executed.

TP Monitor Client Workstation Name

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Basic
Resettable	No	
Element Name	tpmon_client_wkstn	
Element Type	information	
Related Information	• “TP Monitor Client User ID” on page 268 • “TP Monitor Client Application Name” on page 269 • “TP Monitor Client Accounting String” on page 269	

Description: Identifies the client’s system or workstation (for example CICS EITERMID), if the **sqleseti** API was issued in this connection.

Usage: Use this element to identify the user’s machine by node ID, terminal ID, or similar identifiers.

TP Monitor Client Application Name

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Basic
Resettable	No	
Element Name	tpmon_client_app	
Element Type	information	
Related Information	• "TP Monitor Client User ID" on page 268 • "TP Monitor Client Workstation Name" on page 268 • "TP Monitor Client Accounting String" on page 269	

Description: Identifies the the server transaction program performing the transaction, if the **sqleseti** API was issued in this connection.

Usage: Use this element for problem determination and accounting purposes.

TP Monitor Client Accounting String

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Basic
Resettable	No	
Element Name	tpmon_acc_str	
Element Type	information	
Related Information	• "TP Monitor Client User ID" on page 268 • "TP Monitor Client Workstation Name" on page 268 • "TP Monitor Client Application Name" on page 269	

Description: The data passed to the target database for logging and diagnostic purposes, if the **sqleseti** API was issued in this connection.

Usage: Use this element for problem determination and accounting purposes.

Chapter 4. Event Monitor Output

This chapter explains the contents and format of the trace produced by an event monitor and different options that can be specified on the CREATE EVENT MONITOR statement that can influence the trace. It shows how to program for reading this trace, through the use of code samples.

Output Records

The output of an event monitor is a binary stream of logical data groupings that are exactly the same for both pipe and file event monitors. You can format this trace using the db2evmon productivity tool.

The following table illustrates the different groupings which may appear in the event monitor output. See "Information Available from Event Monitors" on page 21 for a list of events that trigger the writing of event records and "Reading an Event Monitor Trace" on page 281 for more information on output. Records in a trace can be divided into four types:

1. Monitor information - identifies the version level of the event monitor.
2. Prologue information - generated when an event monitor is activated.
3. Actual content information - generated as events occur.
4. Epilog information - generated when a database is deactivated.

Event type	Logical data group	Information returned
	Monitor	
Monitor Level	event_log_stream_header (SQLM_EVENT_LOG_STREAM_HEADER)	Identifies the version level and byte order of the event monitor. Applications can use this header to determine whether they can handle the evmon output stream.
	Prolog	
Log Header	log_header_event (SQLM_ELM_EVENT_LOG_HEADER)	Characteristics of the trace, for example server type and memory layout.
Database Header	db_header_event (SQLM_ELM_EVENT_DB_HEADER)	Database name, path and activation time.
Event Monitor Start	start_event (SQLM_ELM_EVENT_START)	Time when the monitor was started or restarted.
Connection Header	connheader_event (SQLM_ELM_EVENT_CONNHEADER)	One for each current connection, includes connection time and application name.
	Actual Contents	
Statement Event	stmt_event (SQLM_ELM_EVENT_STMT)	Statement level data, including text for dynamic statements.

Event type	Logical data group	Information returned
Subsection Event	subsection_event (SQLM_ELM_EVENT_SUBSECTION)	Subsection level data.
Transaction Event	xaction_event (SQLM_ELM_EVENT_XACT)	Transaction level data.
Connection Event	conn_event (SQLM_ELM_EVENT_CONN)	Connection level data.
Deadlock Event	deadlock_event (SQLM_ELM_EVENT_DEADLOCK)	Deadlock level data.
Deadlocked Connection Event	dlconn_event (SQLM_ELM_EVENT_DLCONN)	One for each connection involved in the deadlock, includes applications involved and locks in contention.
Overflow	overflow_event (SQLM_ELM_EVENT_OVERFLOW)	Number of records lost - generated when writer cannot keep up with a (non-blocked) event monitor.
	Epilog	
Database Event	db_event (SQLM_ELM_EVENT_DB)	Database level data.
Buffer Pool Event	bufferpool_event (SQLM_ELM_EVENT_BUFFERPOOL)	Buffer pool level data.
Table Space Event	tablespace_event (SQLM_ELM_EVENT_TABLESPACE)	Table space level data.
Table Event	table_event (SQLM_ELM_EVENT_TABLE)	Table level data.

Event records may be generated for any connection and may therefore appear in mixed order in the stream. This means that you may get a transaction event for Connection 1, immediately followed by a connection event for Connection 2. However, records belonging to a single connection or a single event, will appear in their logical order. For example, a statement record (end of statement) always precedes a transaction record (end of UOW), if any. Similarly, a deadlock event record always precedes the deadlocked connection event records for each connection involved in the deadlock. The **application id** or **application handle (agent_id)** can be used to match records with a connection.

For example, using the following event monitor,

```
db2 connect to sample
db2 "create event monitor ALL for
  statements, transactions, connections,
  deadlocks, database, bufferpools,
  tablespaces, tables, write to
  file '/tmp/all'"
mkdir /tmp/all
db2 connect reset
```

the following workload,

Application 1

```
db2 set event monitor ALL state 1
db2 select evmonname from
  syscat.eventmonitors
db2 connect reset
```

Application 2

```
db2 connect to sample

db2 +c connect reset
```

the following trace might be generated. Listed in this sample are some of the fields in each event record to give a flavor of the type of information contained in a trace. See “Event Monitors” on page 12 for an example of deadlock events. Note, the numbers in this sample are used to illustrate the order in which records have been written.

MONITOR

The Monitor information is generated for all event monitors. Only event monitors that return a version of SQLM_DBMON_VERSION6 use the self-describing data stream.

Pre-Version 6 output must be read using the Version 5 method. For information on these static sized structures refer to the *sqlmon.h* file.

PROLOG

The Prolog information is generated when `set event monitor all state 1` is executed. If this event monitor had been AUTOSTART, it would have been generated when the database was activated.

```
1) event_log_stream_header
      byte_order:         SQLM_BIG_ENDIAN      - a UNIX or AIX box
      size:               400                  - not used, for compatibility only
      version:            SQLM_DBMON_VERSION6  - trace was produced by UDB V6

2) log_header_event
      version:            SQLM_DBMON_VERSION6  - Trace was produced by UDB V6
      num_nodes_in_db2_instance: 1             - for a standalone system,
      byte_order:         SQLM_BIG_ENDIAN      - on a UNIX or AIX box,
      event_monitor_name: ALL                  - by event monitor: 'ALL'

3) dbheader_event
      db_name:            SAMPLE               - for database 'SAMPLE'

4) connheader_event
      agent_id: 14                             - Application 1 - handle
      appl_id:  *LOCAL.bourbon.970602180712    - Application 1 - id with timestamp
```

CONTENTS

Generated when Application 1 issues `select name from syscat.eventmonitors`. At the time that the event monitor is turned on, Application 2 has not yet connected.

```
5) stmt_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      operation:         SQLM_PREPARE
      package_name:      SQLC2BA4
      cursor:            SQLCUR201
      @stmt_text_offset: SELECT EVMONNAME FROM SYSCAT.EVENTMONITORS

6) stmt_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      operation:         SQLM_OPEN
      package_name:      SQLC2BA4
      cursor:            SQLCUR201
      @stmt_text_offset: SELECT EVMONNAME FROM SYSCAT.EVENTMONITORS

7) stmt_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      operation:         SQLM_FETCH
      package_name:      SQLC2BA4
      cursor:            SQLCUR201
      @stmt_text_offset: SELECT EVMONNAME FROM SYSCAT.EVENTMONITORS
      fetch_count:       2
      sqlca.sqlcode:     100  - (all rows in the SYSCAT.EVENTMONITORS table)
      SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of a
      query is an empty table.  SQLSTATE=02000
NOTE - A fetch event is generated only if the fetch fails or encounters end of table

8) stmt_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      operation:         SQLM_DESCRIBE
      package_name:      SQLC2BA4
      cursor:            SQLCUR201
      @stmt_text_offset: SELECT EVMONNAME FROM SYSCAT.EVENTMONITORS

9) stmt_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      operation:         SQLM_CLOSE
      package_name:      SQLC2BA4
      cursor:            SQLCUR201
      @stmt_text_offset: SELECT EVMONNAME FROM SYSCAT.EVENTMONITORS
      fetch_count:       2

10) stmt_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      operation:    SQLM_STATIC_COMMIT       - generated by CLP after the SELECT
      package_name: SQLC2BA4

11) xaction_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      status:    SQLM_UOWCOMMIT
      rows_read: 7
```

Application 2 is connecting to the database. Output is interleaved, as the DB2 agents are executing simultaneously:

```
12) connheader_event
      agent_id: 15                            - Application 2 - handle
      appl_id:  *LOCAL.bourbon.970602180714    - Application 2 - id with timestamp

13) stmt_event
      agent_id: 15
      appl_id:  *LOCAL.bourbon.970602180714
      operation:  SQLM_STATIC_COMMIT          - generated by CLP on CONNECT

14) xaction_event
      agent_id: 15
      appl_id:  *LOCAL.bourbon.970602180714
      status:    SQLM_UOWCOMMIT

15) stmt_event
      agent_id: 15
      appl_id:  *LOCAL.bourbon.970602180714
      operation:  SQLM_STATIC_COMMIT          - generated on CONNECT RESET

16) xaction_event
      agent_id: 15
      appl_id:  *LOCAL.bourbon.970602180714
      status:    SQLM_UOWCOMMIT

17) conn_event
      agent_id: 15
      appl_id:  *LOCAL.bourbon.970602180714
      commit_sql_stmts:    2

18) stmt_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      operation:    SQLM_STATIC_COMMIT       - generated on CONNECT RESET
      package_name: SQLC2BA4

19) xaction_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      status:    SQLM_UOWCOMMIT
      rows_read: 2
      locks_held_top: 7

20) conn_event
      agent_id: 14
      appl_id:  *LOCAL.bourbon.970602180712
      select_sql_stmts: 1
      rows_selected:    2
```

EPILOG

The Epilog information is generated during database deactivation (last application finished disconnecting):

```
21) table_event
      table_schema: SYSIBM
      table_name:   SYSTABLES
      table_type:   SQLM_CATALOG_TABLE
      rows_read: 2

22) table_event
      table_schema: SYSIBM
      table_name:   SYSDBAUTH
      table_type:   SQLM_CATALOG_TABLE
      rows_read: 3

23) tablespace_event
      tablespace_name: SYSCATSPACE

24) tablespace_event
      tablespace_name: TEMPSPACE1

25) tablespace_event
      tablespace_name: USERSPACE1

26) bufferpool_event
      bp_name: IBMDEFAULTBP

27) db_event
      connections_top: 2
```

Note: A WHERE clause on the CREATE EVENT MONITOR SQL statement can be used to restrict the applications that will generate events; see "Appendix A. Database System Monitor Interfaces" on page 299 for details.

Matching Event Records with Their Application

Each record includes the application handle and application ID. These allow you to correlate each record with the application for which the record was generated.

The application handle (**agent_id**) is unique system-wide for the duration of the application. However, it will eventually be reused (a 16 bit counter is used to generate this identifier). In most cases, this reuse is not a problem, since an application reading records from the trace is able to detect a connection that was terminated. For example, encountering (in the trace) a connection header with a known agent_ID implies that the previous connection with this agent_ID was terminated.

The application ID is a string identifier that includes a timestamp and is guaranteed to remain unique, even after stopping and restarting the database manager.

File Event Monitor Buffering

The event monitor output thread buffers records, using two internal buffers, before writing them to disk. Records are written to the trace only when a buffer is full. To force an event monitor to flush its buffers you must turn it off or empty the buffers using the FLUSH EVENT MONITOR command. The size of these buffers can be specified on the CREATE EVENT MONITOR statement with the BUFFERSIZE argument. Specifying larger buffers reduces the number of disk accesses and for event monitors with a high amount of throughput, improves monitoring performance.

Figure 4 illustrates how event records are generated for a FILE statement event monitor: 2 applications are connected to a database, each having a single agent working on its behalf.

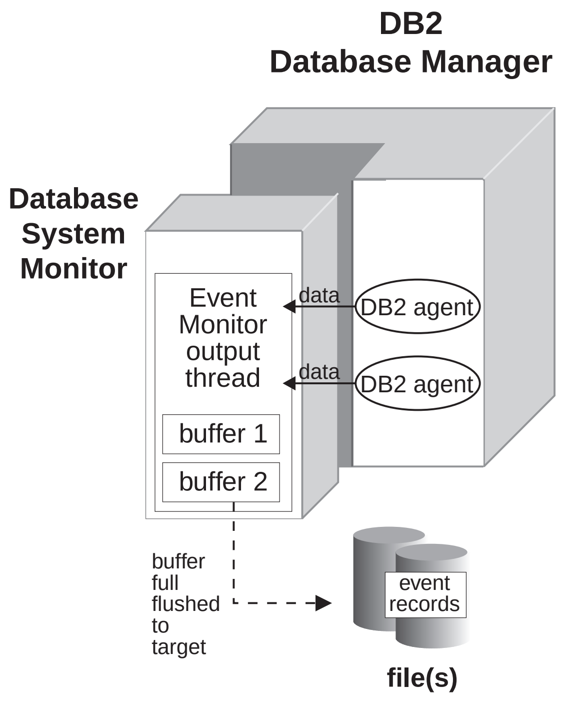


Figure 4. Event Monitor Buffers

In this example, each application agent has just finished executing a statement and is reporting the monitor data it has collected for its statement to the event monitor output thread. The output thread formats the records and writes them into one of its two buffers. The buffer gets written to a file when it is full. Having two buffers allows the output thread to continue receiving data from database agents, while a buffer is being written.

Blocked Event Monitors

A blocked event monitor will suspend the agent(s) sending monitor data, when both of its buffers are full, until a buffer has been written. This can introduce a significant performance overhead, depending on the type of workload and the speed of the I/O device. But, a blocked event monitor never discards event records, as long as it is running. This is the default.

Non-Blocked Event Monitors

A non-blocked event monitor will simply discard monitor data coming from the agents when the data is coming faster than it can write it. This allows event monitoring with less impact on other database activities. The following is sample DDL for creating a non-blocked event monitor:

```
db2 "create event monitor STMT for
statements write to file '/tmp/all'
NONBLOCKED"
```

Overflows

An event monitor that has discarded event records generates an **overflow event**. It specifies the start and stop time during which the monitor was discarding events, and the number of events that were discarded during that period.

Unwritten Overflow Data: It is possible for an event monitor to terminate or be deactivated with a pending overflow to report. If this occurs, the following message is written to the db2diag.log:

```
DIA1603I Event Monitor monitor-name had a pending overflow
record when it was deactivated.
```

File Event Monitor Target

All the output of the event monitor goes in the directory supplied to the FILE argument on the CREATE EVENT MONITOR statement.

When a file event monitor is first activated, a control file is created in this directory. This binary file contains control information that is used to prevent two event monitors from simultaneously writing to the same target, and to keep track of the file and file location where the event monitor is supposed to write its next record. It is named *db2event.ctl*; do not remove or modify this file.

Limiting Trace Size

By default, an event monitor writes its trace to a single file, called *00000000.evt*. This file will keep growing as long as there is space on the file system. You can limit the maximum size of a trace using the MAXFILESIZE and MAXFILES arguments of the create event monitor statement.

Number of Files: The trace produced by an event monitor can be quite large, and you may want to break it down into several files of a fixed size. This also allows you to remove files after processing them, while the event monitor is still running.

Files are numbered sequentially, beginning with *00000000.evt*. If you are using several files, then when a file is full, output is automatically directed to the next file. For example, the following event monitor will break down its trace into 4MB files. It keeps creating files as long as there is space on the file system.

```
db2 "create event monitor BIGONE
for statements, transactions, connections,
deadlocks write to file '/tmp/bigevmon'
MAXFILESIZE 1000
MAXFILES NONE"
```

This might result in the following files in its target directory.

```
File                              size (bytes)
/tmp/bigevmon/db2evmon.ctl           300
/tmp/bigevmon/00000000.evt         4079766
/tmp/bigevmon/00000001.evt         4095128
/tmp/bigevmon/00000002.evt         4095602
```

The highest numbered file is always the active file. When the number of files reaches the maximum defined by MAXFILES, the event monitor deactivates itself and the following message is written to the DB2DIAG.LOG.

```
DIA1601I Event Monitor monitor-name was deactivated when
it reached its preset MAXFILES and MAXFILESIZE limit
```

You can avoid this situation by removing full files (see “Processing Data While Monitor is Active”). Any event file except the active file can be removed while the event monitor is still running.

Running out of Disk Space

When a File event monitor runs out of disk space, it shuts itself down, after logging a system-error-level message in the error logs, *db2diag.log* and *db2err.log*.

Processing Data While Monitor is Active

You may want an event monitor to collect data continuously so that no events are ever missed. For example, if you have a usage account system that uses an event monitor to collect data, you may want to process the data each night beginning at 2:00 AM, at which point you delete the files that have been processed.

An event monitor cannot be forced to switch to the next file unless you stop and restart it. It must also be in APPEND mode. In order to keep track of which events have been processed in the active file, you can create an application that simply keeps track of the file number and location of the last record processed. When processing the trace the next time around, the application can simply seek to that file location.

Using Pipe event monitors is an easy way to read data produced by an active event monitor (see “Using Pipe Event Monitors” on page 22).

Restarting a File Event Monitor

When a File event monitor is restarted, it can either erase any existing data, or append to it.

An APPEND event monitor starts writing at the end of the file it was last using (the file number is indicated in the *db2evmon.ctl* control file). If you have removed that file, then the next file number in sequence is used. For example, in the example above, if you remove all **.evt** files, and restart the event monitor, then event records will be written into *00000003.evt*. If you had not removed the files, then they would go into or append to *00000002.evt*. When an append event monitor is restarted, only the start_event is generated. The event log header and database header are only generated for the first activation.

A REPLACE event monitor always deletes existing event files, and starts writing at *00000000.evt*.

Reading an Event Monitor Trace

Version 6 event monitors return their data in a self-describing format. Figure 5 shows the structure of the data stream and Table 1 on page 282 provides some examples of the logical data groups and data elements that could be returned.

Note: In the examples and tables descriptive names are used for the identifiers. These names are prefixed by **SQLM_ELM_** in the actual data stream. For example, db_event would appear as SQLM_ELM_DB_EVENT in the event monitor output. Types are prefixed with **SQLM_TYPE_** in the actual data stream. For example, headers appear as SQLM_TYPE_HEADER in the data stream.

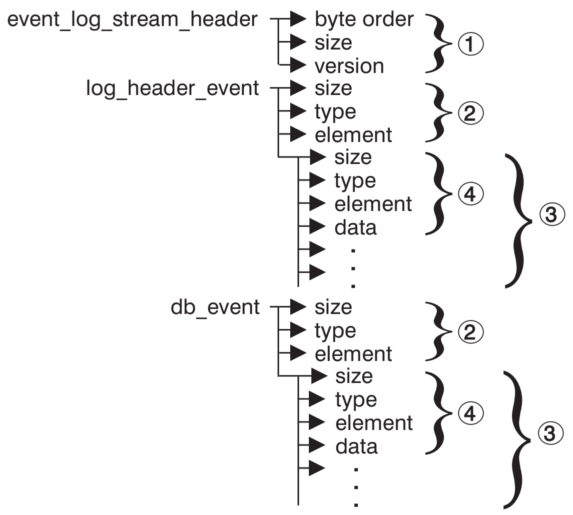


Figure 5. Event Monitor Data Stream

1. The structure of the sqlm_event_log_data_stream_header is different than the other headers in the data stream. The version field determines if the output can be processed as a Version 6 data stream.

   This header has the same size and type as pre-Version 6 event monitor streams. This allows applications to determine if the event monitor output is self-describing or is in the old static format.

Note: This data element is extracted by reading sizeof(sqlm_event_log_data_stream) bytes from the data stream.

2. Each logical data group begins with a header that indicates its size and element name.
3. The size element in the header indicates the size of all the data in that logical data group.
4. Data element information follows its logical data group header and is also self-describing.

Table 1. Sample Event Data Stream

Logical Data Group	Data Stream	Description
event_log_ stream_header	sqlm_little_endian	Byte order of the event monitor data returned.
	1000	Not used (for compatibility with previous releases).
	sqlm_dbmon_version6	The version of the database manager that returned the data. Only version 6 monitors can write data in the self-describing format.
log_header_event	100	Size of the logical data group.
	header	Indicates the start of a logical data group.
	log_header_event	Name of the logical data group.
	4	Size of the data stored in this data element.
	u32bit	Data element type - 32 bit numeric.
	byte_order	The name of the data element collected.
	little_endian	The collected value for this element.
	2	Size of the data stored in this data element.
	u16bit	Data element type - unsigned 16 bit numeric.
	codepage_id	The name of the data element collected.
	850	The collected value for this element.
db_event	100	Size of the logical data group.
	header	Indicates the start of a logical data group.
	db_event	Name of the logical data group.
	4	Size of the data stored in this data element.
	u32bit	Data element type - unsigned 32 bit numeric.
	lock_waits	The name of the data element collected.
	2	The collected value for this element.

Reading the Log Stream Header

The event_log_stream_header identifies the version of the database manager that returned the data. Only Version 6 monitors write their data in the self-describing format.

If you are working with a Version 6 monitor, then you can start processing the self-describing data stream. When reading the trace, you can use the *size* element to skip a logical data group in the trace.

An event monitor, unlike a snapshot monitor (see “Snapshot Output” on page 291), does not have a *size* element that returns the total size of the trace. You typically read an event monitor trace until you reach an end of file.

Reading the Log Header

The log header describes the characteristics of the trace, containing information such as the memory model (for example little endian) of the server where the trace was collected, and the codepage of the database. You may have to do byte swapping on numerical values, if the system where you read the trace has a different memory model than the server (for example, Windows NT to UNIX). Codepage translation may also need to be done, if the database is configured in a different language than the machine from which you read the trace.

The following code can be used to read a single record from the event monitor trace.

```
//-------------------------------------------------------------------------------
// Read an event record  - returns:  0 (success) or EOF
// NOTE: This works for all records except sqlm_event_log_stream_header
//-------------------------------------------------------------------------------
int read_event_record(EventLog *evtlog, char *buffer)
{
   sqlm_header_info*  pHeader = (sqlm_header_info*) buffer;

   //----------------------------------------------------------------------------
   // Read the record header
   //----------------------------------------------------------------------------
   int rc;
   rc=read_data(evtlog, (char *) pHeader, sizeof(sqlm_header_info));
   if (rc)
      return rc;  /* could be at EOF */

   if (evtlog->needByteReversal)
      swapBytes_sqlm_event_rec_header(pHeader);

   //----------------------------------------------------------------------------
   // Read the rest of the data
   //----------------------------------------------------------------------------
   rc=read_data(evtlog, buffer       + sizeof(sqlm_header_info),
                        pHeader->size);

   if (rc==0 && evtlog->needByteReversal)
      swapBytes(pHeader->type, buffer);

   return rc;
} /* end of read_event_record */
```

Reading the Data Stream

The following routines illustrate how you can open, read, or skip bytes from a PIPE or FILE on a UNIX platform.

```
//--------------------------------------------------------------------------------
// File functions  - Using the ANSI C library
//--------------------------------------------------------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
//--------------------------------------------------------------------------------
FILE* openFile(char *file_name) {
   return fopen(file_name,"rb"); /* returns NULL if file cannot be opened */
}
//--------------------------------------------------------------------------------
int closeFile(FILE* handle) {
  return fclose(handle);
}
//--------------------------------------------------------------------------------
int readFromFile(char* buffer, int size, FILE* fp) {
   int rc=0;                           // returns 0 (success); EOF; or errno
   int records_read = fread(buffer, size, 1, fp);
   if (records_read != 1) {
      if (feof(fp))
         rc = EOF;
      else rc = errno;
   } /* end if no data was returned */
   return rc;
} /* end readFromFile */

//--------------------------------------------------------------------------------
// Pipe functions  -  for AIX
//--------------------------------------------------------------------------------
#include <unistd.h>    /* for pipe functions on AIX */
#include <fcntl.h>     /* for definition of O_RDONLY and open() */
//--------------------------------------------------------------------------------
int openNamedPipe (char   *pipe_name) {
   return open(pipe_name, O_RDONLY);
}
//--------------------------------------------------------------------------------
int closeNamedPipe (int handle) {
  return close(handle);
}
//--------------------------------------------------------------------------------
int readFromPipe(int handle, char* buffer, int size) {
   int rc=0;
   int num_bytes;
   num_bytes = read(handle, buffer, size);
   if (num_bytes != size) {
      if (num_bytes==0)
         rc=EOF;
      else rc = num_bytes;
   } /* end did not get the expected number of bytes back from read() */
   return rc;
} /* end readFromPipe */

//-----------------------------------------------------------------------------
// Read data from Event Monitor trace (FILE or PIPE) returns 0 (success) or EOF
//-----------------------------------------------------------------------------
int read_data(EventLog* evtlog,
              char*     buffer,
              int       size) {
   int rc=0;
   if (evtlog->type == EVMFile) {
      rc = readFromFile(buffer, size, evtlog->current_fp);
      if (rc && rc!=EOF) {
          fprintf(stderr, "ERROR: Could not read from: %s\n",
                       evtlog->current_fn);
          exit(1);
      } /* end cannot read the log header from the file */
   } /* end if the Event Monitor Log is read from a file */
   else {
      rc = readFromPipe(evtlog->handle, buffer, size);
      if (rc && rc!=EOF) {
         fprintf(stderr, "ERROR: Could not read a data from: %s\n",
                          evtlog->target);
         exit(2);
      } /* end cannot read from the pipe */
   } /* end else the Event Log is read from a pipe */
   return rc;
} /* end of read_data */

//-----------------------------------------------------------------------------
// Skip n bytes from current position in the trace
//-----------------------------------------------------------------------------
void skip_data(EventLog* evtlog, int n) {
   if (evtlog->type == EVMFile)
      fseek(evtlog->current_fp, n, SEEK_CUR);
   else if (evtlog->type == EVMPipe) {
      lseek(evtlog->handle, n, SEEK_CUR);
   } /* end else pipe event monitor */
} /* end skip_data *//
```

Swapping Bytes in Numerical Values

This code is required when transferring data between systems using different conventions for storing numerical values (for example, UNIX to Windows NT).

```
#include <sqlmon.h>   // DB2 Database Monitor interface
//-----------------------------------------------------------------------------
// Byte conversion macros
//-----------------------------------------------------------------------------
#define SWAP2(s) ((((s) >> 8) & 0xFF) | (((s) << 8) & 0xFF00))

#define SWAP4(l) ((((l) >> 24) & 0xFF) | ((((l) & 0xFF0000) >> 8) & 0xFF00) \
                      | (((l) & 0xFF00) << 8) | ((l) << 24))
```

```
//--------------------------------------------------------------------------------
void swapBytes_sqlm_event_log_stream_header(sqlm_event_log_stream_header* r) {
   r->size =                       SWAP4(r->size);
   r->version =                    SWAP4(r->version);
} // end of swapBytes_sqlm_event_log_header)
```

Printing Event Records

All timestamps in event monitor records are returned in two unsigned 4 byte data elements (seconds and microsec). These represent the GMT time since January 1, 1970.

All strings in event monitor records are padded with blanks, up to their maximum size. **Strings returned by DB2 are NOT NULL TERMINATED.**

Chapter 5. Snapshot Monitor Output

This chapter explains the contents and format of the information captured by a snapshot. It shows how to program for working with the snapshot monitor and how to parse through the self-describing output.

Snapshot Requests

The following table lists the Snapshot request types and the logical data groupings that can be returned. Figure 6 on page 290 shows the hierarchical structure of the logical data groups.

API request type	Logical data groupings that may be returned	Information returned
All Snapshots	collected (SQLM_ELM_COLLECTED)	Information relevant to the entire snapshot. Including an indication of the number of lower level logical data groups that follow.
SQLMA_DB2	db2 (SQLM_ELM_DB2)	DB2 instance information.
	fcm (SQLM_ELM_FCM)	FCM information.
	fcm_node (SQLM_ELM_FCM_NODE)	FCM node information.
SQLMA_APPLINFO_ALL SQLMA_DBASE_APPLINFO	appl_info (SQLM_ELM_APPL_INFO)	Application identification information.
SQLMA_DCS_APPLINFO_ALL	dcs_appli_info (SQLM_ELM__DCS_APPL_INFO)	DCS application identification information.
SQLMA_DCS_APPL SQLMA_DCS_APPL_HANDLE SQLMA_DCS_DBASE_APPLS	dcs_appl (SQLM_ELM_DCS_APPL)	DCS application information.
	dcs_stmt (SQLM_ELM_DCS_STMT)	DCS statement information.
	dcs_appl_info (SQLM_ELM_DCS_APPL_INFO)	DCS application identification information.

API request type	Logical data groupings that may be returned	Information returned
SQLMA_APPL SQLMA_AGENT_ID SQLMA_DBASE_APPLS	appl (SQLM_ELM_APPL)	Application information.
	agent (SQLM_ELM_AGENT)	Agent information.
	appl_info (SQLM_ELM_APPL_INFO)	Application information.
	lock_wait (SQLM_ELM_LOCK_WAIT)	Lock waiting information.
	stmt (SQLM_ELM_STMT)	Statement information.
	subsection (SQLM_ELM_SUBSECTION)	Subsection information.
	agent (SQLM_ELM_AGENT)	Subagent information for parallel SQL processing in partitioned databases and on SMP machines.
SQLMA_DCS_DBASE SQLMA_DCS_DBASE_ALL	dcs_dbase (SQLM_ELM_DCS_DBASE)	DCS database information.
SQLMA_DBASE SQLMA_DBASE_ALL	dbase (SQLM_ELM_DBASE)	Database information.
	rollforward (SQLM_ELM_ROLLFORWARD)	Rollforward information.
	tablespace (SQLM_ELM_TABLESPACE)	Tablespace information
SQLMA_DBASE_TABLES	table_list (SQLM_ELM_TABLE_LIST)	Table information.
	tables (SQLM_ELM_TABLE)	Database-wide table information.
SQLMA_APPL_LOCKS SQLMA_APPL_LOCKS_AGENT_ID	appl_lock_list (SQLM_ELM_APPL_LOCK_LIST)	A listing of application locks.
	lock_wait (SQLM_ELM_LOCK_WAIT)	If any lock waits, they precede locks.
	lock (SQLM_ELM_LOCK)	Lock information.
SQLMA_DBASE_LOCKS	db_lock_list (SQLM_ELM_DB_LOCK_LIST)	A listing of database locks.
	lock_wait (SQLM_ELM_LOCK_WAIT)	If any lock waits, they preceed locks.
	lock (SQLM_ELM_LOCK)	Lock information.
	appl_lock_list (SQLM_ELM_APPL_LOCK_LIST)	A listing of application locks.
SQLMA_DBASE_TABLESPACES	tablespace_list (SQLM_ELM_TABLESPACE_LIST)	Database-wide tablespace information.
	tablespace (SQLM_ELM_TABLESPACE)	Tablespace information.
SQLMA_BUFFERPOOLS_ALL SQLMA_DBASE_BUFFERPOOLS	bufferpool (SQLM_ELM_BUFFERPOOL)	Buffer pool information.

API request type	Logical data groupings that may be returned	Information returned
SQLMA_DYNAMIC_SQL	dysql_list (SQLM_ELM_DYNSQL_LIST)	List of dynamic SQL statements.
	dysql (SQLM_ELM_DYNSQL)	Dynamic SQL statement information.

The following figure shows the order that logical data groupings may appear in a snapshot data stream.

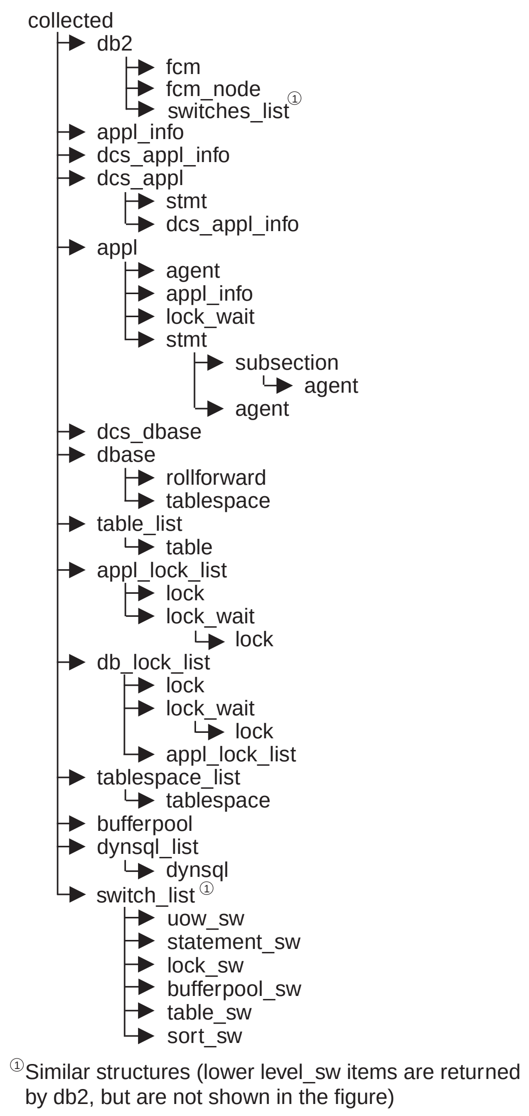


Figure 6. Data Stream Hierarchy

Note: Times may be returned as part of any logical data grouping.

Snapshot Output

A Version 6 snapshot monitor returns its data as a self-describing data stream. Figure 7 shows the structure of the data stream and Table 2 on page 292 provides some examples of the logical data groups and data elements that may be returned.

Note: In the examples and tables descriptive names are used for the identifiers. These names are prefixed by **SQLM_ELM_** in the actual data stream. For example, collected would appear as SQLM_ELM_COLLECTED in the snapshot monitor output. Types are prefixed with **SQLM_TYPE_** in the actual data stream. For example, headers appear as SQLM_TYPE_HEADER in the data stream.

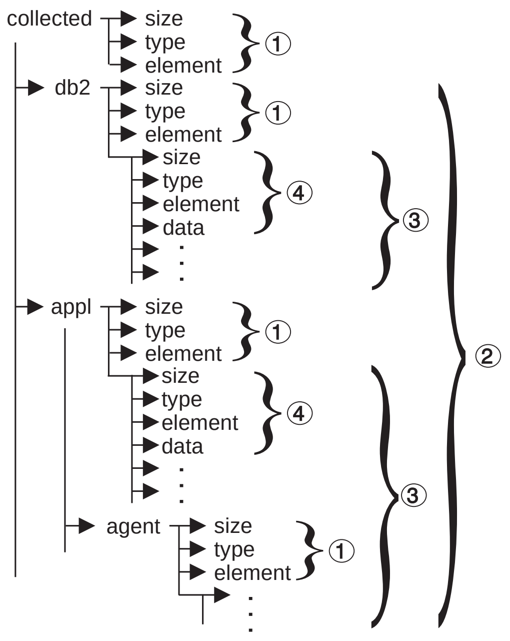


Figure 7. Snapshot Monitor Data Stream

1. Each logical data group begins with a header that indicates its size and name.
2. Size in the collected header returns the total size of the snapshot.
3. The size element in other headers indicates the size of all the data in that logical data group, including any subordinate groupings.
4. Data element information follows its logical data group header and is also self-describing.

Table 2. Sample Snapshot Data Stream

Logical Data Group	Data Stream	Description
collected	1000	Size of the entire snapshot buffer (in bytes).
	header	Indicates the start of a logical data group.
	collected	Name of the logical data group.
	4	Size of the data stored in this data element.
	u32bit	Data element type - unsigned 32 bit numeric.
	server_db2_type	The name of the data element collected.
	sqlf_nt_server	The collected value for this element.
	2	Size of the data stored in this data element.
	u16bit	Data element type - unsigned 16 bit numeric.
	node_number	The name of the data element collected.
	3	The collected value for this element.
db2	200	Size of the db2 level portion of data in the snapshot.
	header	Indicates the start of a logical data group.
	db2	Name of the logical data group.
	4	Size of the data stored in this data element.
	u32bit	Data element type - unsigned 32 bit numeric.
	sort_heap_allocated	The name of the data element collected.
	16	The collected value for this element.
	4	Size of the data stored in this data element.
	u32bit	Data element type - unsigned 32 bit numeric.
	local_cons	The name of the data element collected.
	3	The collected value for this element.
appl	100	Size of the appl element data in the snapshot.
	header	Indicates the start of a logical data group.
	appl	Name of the logical data group.
	4	Size of the data stored in this data element.
	u32bit	Data element type - unsigned 32 bit numeric.
	locks_held	The name of the data element collected.
	3	The collected value for this element.
agent	50	Size of the agent portion of the appl structure.
	header	Indicates the start of a logical data group.
	agent	Name of the logical data group.
	4	Size of the data stored in this data element.
	u32bit	Data element type - 32 bit numeric.
	agent_pid	The name of the data element collected.
	12	The collected value for this element.

Snapshot Scenarios

The following table lists the snapshot scenarios available with Version 6 clients.

Note: From Version 5 clients, binary compatibility is maintained and the lowest down-level server that can be attached to is Version 5.

Table 3. Client/Server Snapshot Scenarios

Snapshot Version Requested	Server Version	Data Format Returned	Action
SQLM_DBMON_VERSION1 SQLM_DBMON_VERSION2 SQLM_DBMON_VERSION5 SQLM_DBMON_VERSION5_2	DB2 Version 5 through Version 6	fixed size structures	Parse the data stream using the fixed structure method.
SQLM_DBMON_VERSION6	DB2 Version 6	self-describing	Parse using the methods described in this chapter (see "Reading the Snapshot" on page 295). The db2ConvMonStream() API can be used to make migrating existing monitor applications easier (see "db2ConvMonStream" on page 310).
SQLM_DBMON_VERSION6	DB2 Version 5	fixed size structures	Parse the data stream using the fixed structure method.

Making a Snapshot Request

An invocation of db2GetSnapshot() can specify several requests (if desired).

```
/* Get Snapshot Data Interface Structure                                      */
typedef struct
{
        sqlma *               piSqlmaData; /* Pointer to monitor area         */
        sqlm_collected *      poCollectedData; /* Pointer to collected data  */
        void                  *poBuffer; /* Pointer to output buffer          */
        db2Uint32             iVersion; /* Snapshot version                   */
        db2Uint32             iBufferSize; /* Size of output buffer           */
        db2Uint32             iStoreResult; /* Write to file flag             */
} db2GetSnapshotData;



SQL_API_RC SQL_API_FN                 /* Get snapshot                         */
  db2GetSnapshot (
        db2Uint32 versionNumber,      /* Database version number              */
        void * pParamStruct,          /* In/out parameters                    */
        struct sqlca * pSqlca);       /* SQLCA                                */
```

The *sqlma* supplied as input argument to db2GetSnapshot() contains an array of **sqlm_obj_struct**. Each sqlm_obj_struct is an individual snapshot request.

sqlm_obj_struct is defined as follows:

```
typedef struct sqlm_obj_struct         /* SNAPSHOT REQUEST  */
{
   unsigned long agent_id;             /* used for requests based on agentid */
   unsigned long obj_type;             /* Snapshot Request Type (SQLMA_XXXX) */
   char          object[SQLM_OBJECT_SZ];/* used for requests based on object  */
                               /* name, such as 'get snapshot for database' */
}sqlm_obj_struct;
```

Where *agent_id* and *object* are only required if applicable for the request type, and are mutually exclusive. For example: a database name is required when the type is SQLMA_DBASE_LOCKS (get snapshot for locks on database), whereas an agent_id is required when the type is SQLMA_APPL_LOCKS_AGENT_ID. Both agent_id and object are ignored for requests such as SQLMA_APPLINFO_ALL (list applications).

Note that agent_id is the **application handle** for an application. It does not correspond to any Operating System process Id (it is named this way for source compatibility with older releases of DB2).

The size of strings returned by DB2 are the actual string length. Strings are not NULL terminated.

Setting Up the sqlma and Issuing the Snapshot Call

The following example sets up the sqlma for a call to db2GetSnapshot() that requests two different snapshots. The first request requires an object name, the database alias, the second request requires an agent_id, the application handle:

```
#include "string.h"
#include "stdlib.h"
#include "stdio.h"
#include "sqlutil.h"
#include "sqlmon.h"  // System Monitor interface
#include "db2ApiDf.h"
main()
{
  struct sqlca sqlca;
  int rc;

  db2GetSnapshotData ss_data;

  #define BUFFER_SZ 4096                 // Use a fixed size output buffer
  char snap_buffer[BUFFER_SZ];    // Snapshot output buffer
  sqlm_collected collected;

  //--------------------------------------------------------------------
  // Request SQLMA_DBASE, and SQLMA_APPL_LOCKS_AGENT_ID in the sqlma
  //--------------------------------------------------------------------
  unsigned long agent_id = 0; // STUB: Obtain by issuing 'list application'

  // Allocate the variable size sqlma structure
  struct sqlma* sqlma = (struct sqlma *) malloc(SQLMASIZE(2));

  // Request 2 different snapshots in same call
  sqlma->obj_num = 2;
  sqlma->obj_var[0].obj_type      = SQLMA_DBASE;
  strcpy(sqlma->obj_var[0].object, "SAMPLE");

  sqlma->obj_var[1].obj_type      = SQLMA_APPL_LOCKS_AGENT_ID;
  sqlma->obj_var[1].agent_id      = agent_id;

  //---------------------------------------------------------------------
  // Perform the snapshot
 //---------------------------------------------------------------------

  ss_data.piSqlmaData = sqlma;
  ss_data.poCollectedData = &collected;
  ss_data.poBuffer = snap_buffer;
  ss_data.iVersion = SQLM_DBMON_VERSION6;
  ss_data.iBufferSize = sizeof(snap_buffer);
  ss_data.iStoreResult = FALSE;

  rc = db2GetSnapshot(db2Version610,
                      ss_data,
                      &sqlca);
  if (sqlca.sqlcode < 0) {
      // get and display a printable error message
      char msg[1024];
      sqlaintp (msg, sizeof(msg), 60, &sqlca);
      printf("%s", msg);
  }
  free(sqlma);
  return rc;
}
```

Reading the Snapshot

The db2GetSnapshot() routine returns snapshot data as a self-describing data stream in the user supplied buffer. Data is returned in the logical data groupings described in “Snapshot Requests” on page 287.

Each item returned by a snapshot request contains fields that specify its size and type (see “Snapshot Output” on page 291). The size can be used to parse through the returned data.

Size can also be used to skip over a logical data group. For example, to skip over a db2 record you need to determine

```
  size of the db2 logical data grouping + sizeof(sqlm_header_info)
```

bytes in the data stream.

The following code sample illustrates how an application could parse through the data returned in the snapshot output buffer. The element **datastream** passed into the function is the buffer returned from the db2GetSnapshot() call.

```
#include "stdlib.h"
#include "stdio.h"
#include "sqlutil.h"
#include "string.h"
#include "sqlmon.h"  // System Monitor interface
```

```
void pocess_buffer(sqlm_header_info *datastream)
{
  long data_len    = datastream->size;
  sqlm_header_info *traversal_ptr = datastream;

  // presume that we aren't interested in the "collected" data
  // elements
  ++traversal_ptr;

  //----------------------------------------------------------------------
  // PROCESS EACH RECORD THAT MAY BE RETURNED IN THE SNAPSHOT OUTPUT BUFFER
  //----------------------------------------------------------------------
  while (data_len > 0)
  {
     // Switch on the element
     switch (traversal_ptr->element)
     {
       case SQLM_ELM_DB2:
           // Process the database manager snapshot
           printf("Processing database manager snapshot\n");
           // ...
           break;
       case SQLM_ELM_DBASE:
           // Process the snapshot ...
           printf("Processing database snapshot\n");
           // ...
           break;
       case SQLM_ELM_APPL:
           printf("Processing application snapshot\n");
           // ...
           break;
       case SQLM_ELM_APPL_INFO:
           printf("Processing list application\n");
           // ...
           break;
       case SQLM_DCS_APPL_INFO:
           printf("Processing list dcs application\n");
           // ...
           break;
       case SQLM_ELM_TABLE_LIST:
           printf("Processing list tables\n");
           // ...
           break;
       case SQLM_ELM_DBASE_LOCK_LIST:
           printf("Processing snapshot for locks on database\n");
           // ...
           break;
       case SQLM_ELM_APPL_LOCK: {
           printf("Processing snapshot for locks for application\n");
           // ...
           break;
       case SQLM_ELM_TABLESPACE_LIST: {
           printf("Processing snapshot for tablespaces\n");
           // ...
```

```
            break;
        default:
            // Do nothing. This could be a new logical data element we aren't
            // interested in, or it could be one of the collected data elements.
     } // end check the current snapshot buffer structure

     // Skip the record we just processed
     data_len -= traversal_ptr->size + sizeof(sqlm_header_info);
     traversal_ptr = (sqlm_header_info *)((char *)traversal_ptr +
                          traversal_ptr->size + sizeof(sqlm_header_info));
  } // end while there are top-level structures in the snapshot output buffer
}
```

Processing portions of the new data stream is similar to processing the uppermost portion of the stream. The following is an example of how a user could pick out the db name from a database logical grouping of data elements, to return it in a pre-allocated string.

A similar approach can be taken for processing any data element from the stream.

```
void process_db2_info(sqlm_header_info *db2inf, char *db_name)
{
  long data_size = db2inf->header.size;
  sqlm_header_data *traverse_ptr = (sqlm_header_data *)db2inf;
  traverse_ptr++;

  while(data_size)
  {
    switch(traverse_ptr->header.element)
    {
      case SQLM_ELM_DB_NAME:
        memcpy(db_name, traverse_ptr->data, traverse_ptr->header.size);
        // Add the null terminator
        db_name[traverse_ptr->header.size] = '\0';
      break;

      // cases to access other elements of interest
      // ...
      default:
      break;
    }
    data_size -= (traverse_ptr->header.size + sizeof(sqlm_header_info));
    traverse_ptr = (sqlm_header_data *)((char *)traverse_ptr->header.size +
                       sizeof(sqlm_header_info));
  }
}
```

Appendix A. Database System Monitor Interfaces

Monitoring task	**Interface (API, Command, SQL Statement)**
Activating an event monitor	"SET EVENT MONITOR STATE" on page 351
Converting the new monitor datastream	"db2ConvMonStream" on page 310
Creating an event monitor	"CREATE EVENT MONITOR" on page 300
Deactivating an event monitor	"SET EVENT MONITOR STATE" on page 351
Determining the state of an event monitor	"EVENT_MON_STATE" on page 322
Displaying the database manager monitor switches	"GET DATABASE MANAGER MONITOR SWITCHES" on page 324
Displaying the database system monitor switches	"GET MONITOR SWITCHES" on page 326 "sqlmon - Get/Update Monitor Switches" on page 355
Estimating the size of a snapshot	"sqlmonsz - Estimate Size Required for db2GetSnapshot() Output Buffer" on page 358
Formatting the event monitor trace	"db2evmon - Event Monitor Productivity Tool" on page 315
Listing the active databases	"LIST ACTIVE DATABASES" on page 342
Listing the applications connected to a database	"LIST APPLICATIONS" on page 344
Listing the DCS applications	"LIST DCS APPLICATIONS" on page 346
Removing an event monitor	"DROP EVENT MONITOR Command and SQL" on page 321
Resetting monitor counters	"RESET MONITOR" on page 349 "sqlmrset - Reset Monitor" on page 361
Starting the event analyzer	"db2eva - Event Analyzer" on page 313
Taking a snapshot	"GET SNAPSHOT" on page 328 "db2GetSnapshot - Get Snapshot" on page 317
Updating the database system monitor switches	"UPDATE MONITOR SWITCHES" on page 364 "sqlmon - Get/Update Monitor Switches" on page 355
Viewing SQL statements	"SQLCACHE_SNAPSHOT" on page 353
Writing event monitor values	"FLUSH EVENT MONITOR" on page 323

CREATE EVENT MONITOR

The CREATE EVENT MONITOR statement defines a monitor that will record certain events that occur when using the database. The definition of each event monitor also specifies where the database should record the events.

Scope

This statement can be embedded in an application program or issued interactively. It is an executable statement that can be dynamically prepared. However, if the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared (SQLSTATE 42509).

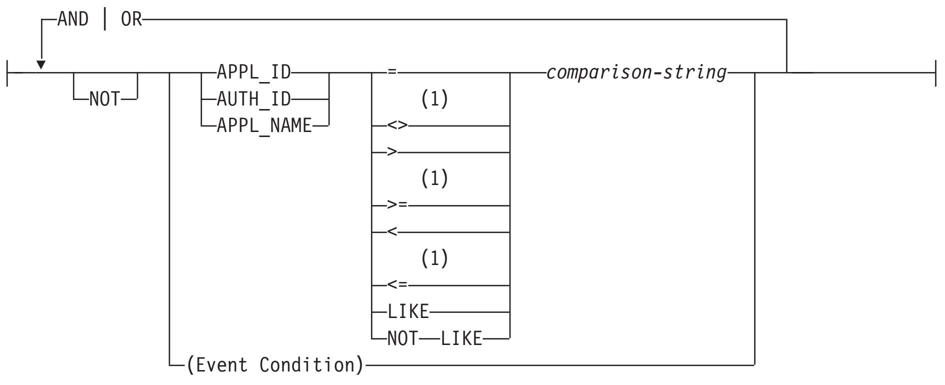
Authorization

The privileges held by the authorization ID must include either SYSADM or DBADM authority (SQLSTATE 42502).

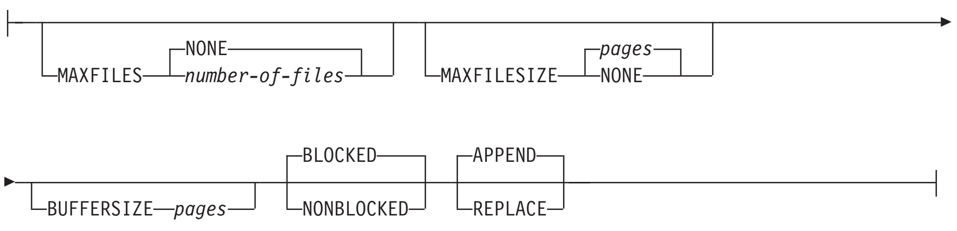
Command Syntax

```
>>--CREATE--EVENT--MONITOR--event-monitor-name--FOR---------------------------->

     .-,-----------------------------------------------------------.
     V                                                             |
>-------+--+-DATABASE----+--------------------------------------+--+---------->
        |  +-TABLES------+                                      |
        |  +-DEADLOCKS---+                                      |
        |  +-TABLESPACES-+                                      |
        |  '-BUFFERPOOLS-'                                      |
        '----+-CONNECTIONS--+--+------------------------------+-'
             +-STATEMENTS---+  '-WHERE--| Event Condition |---'
             '-TRANSACTIONS-'

                                                          .-MANUALSTART-.
>--WRITE--TO--+-PIPE--pipe-name------------------------+--+-------------+--->
              '-FILE--path-name--| File Options |------'  '-AUTOSTART---'

                                   .-LOCAL--.
>--+-------------------------+-----+--------+---------------------------------><
   '-ON NODE--node-number----'     '-GLOBAL-'
```

Event Condition:



File Options:



Notes:

1. Other forms of these operators are also supported. See *SQL Reference* for more details.

Command Parameters

event-monitor-name
: Names the event monitor. This is a one-part name. It is an SQL identifier (either ordinary or delimited). The *event-monitor-name* must not identify an event monitor that already exists in the catalog (SQLSTATE 42710).

FOR
: Introduces the type of event to record.

DATABASE
: Specifies that the event monitor records a database event when the last application disconnects from the database.

TABLES
: Specifies that the event monitor records a table event for each active table when the last application disconnects from the database. An active table is a table that has changed since the first connection to the database.

DEADLOCKS
Specifies that the event monitor records a deadlock event whenever a deadlock occurs.

TABLESPACES
Specifies that the event monitor records a table space event for each table space when the last application disconnects from the database.

BUFFERPOOLS
Specifies that the event monitor records a buffer pool event when the last application disconnects from the database.

CONNECTIONS
Specifies that the event monitor records a connection event when an application disconnects from the database.

STATEMENTS
Specifies that the event monitor records a statement event whenever a SQL statement finishes executing.

TRANSACTIONS
Specifies that the event monitor records a transaction event whenever a transaction completes (that is, whenever there is a commit or rollback operation).

WHERE *event condition*
Defines a filter that determines which connections cause a CONNECTION, STATEMENT or TRANSACTION event to occur. If the result of the event condition is TRUE for a particular connection, then that connection will generate the requested events.

This clause is a special form of the WHERE clause that should not be confused with a standard search condition.

To determine if an application will generate events for a particular event monitor, the WHERE clause is evaluated:

1. For each active connection when an event monitor is first turned on.
2. Subsequently for each new connection to the database at connect time.

The WHERE clause is not evaluated for each event.

If no WHERE clause is specified then all events of the specified event type will be monitored.

APPL_ID
Specifies that the application ID of each connection should be compared with the *comparison-string* in order to determine if the

connection should generate CONNECTION, STATEMENT or TRANSACTION events (whichever was specified).

AUTH_ID
Specifies that the authorization ID of each connection should be compared with the *comparison-string* in order to determine if the connection should generate CONNECTION, STATEMENT or TRANSACTION events (whichever was specified).

APPL_NAME
Specifies that the application program name of each connection should be compared with the *comparison-string* in order to determine if the connection should generate CONNECTION, STATEMENT or TRANSACTION events (whichever was specified).

The application program name is the first 20 bytes of the application program file name, after the last path separator.

comparison-string
A string to be compared with the APPL_ID, AUTH_ID, or APPL_NAME of each application that connects to the database. *comparison-string* must be a string constant (that is, host variables and other string expressions are not permitted).

WRITE TO
Introduces the target for the data.

PIPE
Specifies that the target for the event monitor data is a named pipe. The event monitor writes the data to the pipe in a single stream (that is, as if it were a single, infinitely long file). When writing the data to a pipe, an event monitor does not perform blocked writes. If there is no room in the pipe buffer, then the event monitor will discard the data. It is the monitoring application's responsibility to read the data promptly if it wishes to ensure no data loss.

pipe-name
The name of the pipe (FIFO on AIX) to which the event monitor will write the data.

The naming rules for pipes are platform specific. On UNIX operating systems pipe names are treated like file names. As a result, relative pipe names are permitted, and are treated like relative path-names (see *path-name* below). However, on OS/2, Windows 95 and Windows NT, there is a special syntax for a pipe name. As a result, on OS/2, Windows 95 and Windows NT absolute pipe names are required.

The existence of the pipe will not be checked at event monitor creation time. It is the responsibility of the monitoring application to have created and opened the pipe for reading at the time that the event monitor is activated. If the pipe is not available at this time, then the event monitor will turn itself off, and will log an error. (That is, if the event monitor was activated at database start time as a result of the AUTOSTART option, then the event monitor will log an error in the system error log.) If the event monitor is activated via the SET EVENT MONITOR STATE SQL statement, then that statement will fail (SQLSTATE 58030).

FILE

Indicates that the target for the event monitor data is a file (or set of files). The event monitor writes out the stream of data as a series of 8 character numbered files, with the extension "evt". (for example, 00000000.evt, 00000001.evt, 00000002.evt, etc). The data should be considered to be one logical file even though the data is broken up into smaller pieces (that is, the start of the data stream is the first byte in the file 00000000.evt; the end of the data stream is the last byte in the file nnnnnnnn.evt).

The maximum size of each file can be defined as well as the maximum number of files. An event monitor will never split a single event record across two files. However, an event monitor may write related records in two different files. It is the responsibility of the application that uses this data to keep track of such related information when processing the event files.

path-name

The name of the directory in which the event monitor should write the event files data. The path must be known at the server, however, the path itself could reside on another partition or node (for example, in a UNIX-based system, this might be an NFS mounted file). A string constant must be used when specifying the *path-name*.

The directory does not have to exist at CREATE EVENT MONITOR time. However, a check is made for the existence of the target path when the event monitor is activated. At that time, if the target path does not exist, an error (SQLSTATE 428A3) is raised.

If an absolute path (a path that starts with the root directory on AIX, or a disk identifier on OS/2, Windows 95 and Windows NT) is specified, then the specified path will be the one used. If a relative path (a path that does not start with the root) is specified, then the path relative to the DB2EVENT directory in the database directory will be used.

When a relative path is specified, the DB2EVENT directory is used to convert it into an absolute path. Thereafter, no distinction is made between absolute and relative paths. The absolute path is stored in the SYSCAT.EVENTMONITORS catalog view.

It is possible to specify two or more event monitors that have the same target path. However, once one of the event monitors has been activated for the first time, and as long as the target directory is not empty, it will be impossible to activate any of the other event monitors.

File Options
Specifies the options for the file format.

MAXFILES NONE
Specifies that there is no limit to the number of event files that the event monitor will create. This is the default.

MAXFILES *number-of-files*
Specifies that there is a limit on the number of event monitor files that will exist for a particular event monitor at any time. Whenever an event monitor has to create another file, it will check to make sure that the number of .evt files in the directory is less than *number-of-files*. If this limit has already been reached, then the event monitor will turn itself off.

If an application removes the event files from the directory after they have been written, then the total number of files that an event monitor can produce can exceed *number-of-files*. This option has been provided to allow a user to guarantee that the event data will not consume more than a specified amount of disk space.

MAXFILESIZE *pages*
Specifies that there is a limit to the size of each event monitor file. Whenever an event monitor writes a new event record to a file, it checks that the file will not grow to be greater than *pages* (in units of 4K pages). If the resulting file would be too large, then the event monitor switches to the next file. The default for this option is:

- OS/2, Windows 95 and Windows NT - 200 4K pages
- UNIX - 1000 4K pages

The number of pages must be greater than at least the size of the event buffer in pages. If this requirement is not met, then an error (SQLSTATE 428A4) is raised.

MAXFILESIZE NONE
Specifies that there is no set limit on a file's size. If MAXFILESIZE NONE is specified, then MAXFILES 1 must also be specified. This option means that one file will contain all of the event data for a particular event monitor. In this case the only event file will be 00000000.evt.

BUFFERSIZE *pages*
Specifies the size of the event monitor buffers (in units of 4K pages). All event monitor file I/O is buffered to improve the performance of the event monitors. The larger the buffers, the less I/O will be performed by the event monitor. Highly active event monitors should have larger buffers than relatively inactive event monitors. When the monitor is started, two buffers of the specified size are allocated. Event monitors use double buffering to permit asynchronous I/O.

The minimum and default size of each buffer (if this option is not specified) is 1 page (that is, 2 buffers, each 4 K in size). The maximum size of the buffers is limited by the size of the database heap (DBHEAP) since the buffers are allocated from the heap. If using a lot of event monitors at the same time, increase the size of the DBHEAP database configuration parameter.

Event monitors that write their data to a pipe also have two internal (non-configurable) buffers that are each 1 page in size. These buffers are also allocated from the database heap (DBHEAP). For each active event monitor that has a pipe target, increase the size of the database heap by 2 pages.

BLOCKED
Specifies that each agent that generates an event should wait for an event buffer to be written out to disk if the agent determines that both event buffers are full. BLOCKED should be selected to guarantee no event data loss. This is the default option.

NONBLOCKED
Specifies that each agent that generates an event should not wait for the event buffer to be written out to disk if the agent determines that both event buffers are full. NONBLOCKED event monitors do not slow down database operations to the extent of BLOCKED event

monitors. However, NONBLOCKED event monitors are subject to data loss on highly active systems.

APPEND
Specifies that if event data files already exist when the event monitor is turned on, then the event monitor will append the new event data to the existing stream of data files. When the event monitor is re-activated, it will resume writing to the event files as if it had never been turned off. APPEND is the default option.

The APPEND option does not apply at CREATE EVENT MONITOR time, if there is existing event data in the directory where the newly created event monitor is to write its event data.

REPLACE
Specifies that if event data files already exist when the event monitor is turned on, then the event monitor will erase all of the event files and start writing data to file 00000000.evt.

MANUALSTART
Specifies that the event monitor not be started automatically each time the database is started. Event monitors with the MANUALSTART option must be activated manually using the SET EVENT MONITOR STATE statement. This is the default option.

AUTOSTART
Specifies that the event monitor be started automatically each time the database is started.

ON NODE
Keyword that indicates that specific partitions are specified.

node-number
Specifies a partition number where the event monitor runs and write the events. With the monitoring scope defined as GLOBAL, all partitions report to the specified partition number. The I/O component will physically run on the specified partition, writing its records to /tmp/dlocks direcotry on that partition.

GLOBAL
Event monitor reports from all partitions. For a partitioned database in DB2 Universal Database Version 5.2, only deadlock event monitors can be defined as GLOBAL. The global event monitor will report deadlocks for all nodes in the system.

LOCAL
Event monitor reports only on the partition that is running. It gives a partial trace of the database activity. This is the default.

Sample Programs

- Each of the event types (DATABASE, TABLES, DEADLOCKs,...) can only be specified once in a particular event monitor definition.

Usage Notes

- Event monitor definitions are recorded in the SYSCAT.EVENTMONITORS catalog view. The events themselves are recorded in the SYSCAT.EVENTS catalog view.

Examples

Example 1: The following example creates an event monitor called SMITHPAY. This event monitor, will collect event data for the database as well as for the SQL statements performed by the PAYROLL application owned by the JSMITH authorization ID. The data will be appended to the absolute path /home/jsmith/event/smithpay/. A maximum of 25 files will be created. Each file will be a maximum of 1024 4K pages long. The file I/O will be non-blocked.

```
CREATE EVENT MONITOR SMITHPAY
   FOR DATABASE, STATEMENTS
   WHERE APPL_NAME = 'PAYROLL' AND AUTH_ID = 'JSMITH'
   WRITE TO FILE '/home/jsmith/event/smithpay'
   MAXFILES 25
   MAXFILESIZE 1024
   NONBLOCKED
   APPEND
```

Example 2: The following example creates an event monitor called DEADLOCKS_EVTS. This event monitor will collect deadlock events and will write them to the relative path DLOCKS. One file will be written, and there is no maximum file size. Each time the event monitor is activated, it will append the event data to the file 00000000.evt if it exists. The event monitor will be started each time the database is started. The I/0 will be blocked by default.

```
CREATE EVENT MONITOR DEADLOCK_EVTS
   FOR DEADLOCKS
   WRITE TO FILE 'DLOCKS'
   MAXFILES 1
   MAXFILESIZE NONE
   AUTOSTART
```

Example 3: This example creates an event monitor called DB_APPLS. This event monitor collects connection events, and writes the data to the named pipe /home/jsmith/applpipe.

```
CREATE EVENT MONITOR DB_APPLS
   FOR CONNECTIONS
   WRITE TO PIPE '/home/jsmith/applpipe'
```

db2ConvMonStream

Converts the new, self-describing format for a single logical data element (for example, SQLM_ELM_DB2) to the corresponding pre-version 6 external monitor structure (for example, sqlm_db2). When upgrading API calls to use the post-version 5 stream, one must traverse the monitor data using the new stream format (for example, the user must find the SQLM_ELM_DB2 element). This portion of the stream can then be passed into the conversion API to get the associated pre-version 6 data.

Authorization

None

Required Connection

None

API Include File

db2ApiDf.h

C API Syntax

```
/* File: db2ApiDf.h */
/* API: db2ConvMonStream */
/* ... */
int db2ConvMonStream (
  unsigned char version,
  db2ConvMonStreamData * data,
  struct sqlca * pSqlca);

typedef struct
{
  void * poTarget;
  sqlm_header_info * piSource;
  db2Uint32 iTargetType;
  db2Uint32 iTargetSize;
  db2Uint32 iSourceType
} db2ConvMonStreamData;
/* ... */
```

API Parameters

version
Input. Specifies the version and release level of the structure passed in as the second parameter, *data*.

data Input. A pointer to the *db2ConvMonStreamData* structure.

pSqlca
: Output. A pointer to the *sqlca* structure. For more information about this structure, see the *Administrative API Reference*.

poTarget
: Output. A pointer to the target monitor output structure (for example, sqlm_db2). A list of output types, and their corresponding input types, is given below.

piSource
: Input. A pointer to the logical data element being converted (for example, SQLM_ELM_DB2). A list of output types, and their corresponding input types, is given below.

iTargetType
: Input. The type of conversion being performed. Specify the value for the v5 type in `sqlmon.h` for instance SQLM_DB2_SS.

iTargetSize
: Input. This parameter can usually be set to the size of the structure pointed to by *poTarget*; however, for elements that have usually been referenced by an offset value from the end of the structure (for example, statement text in *sqlm_stmt*), specify a buffer that is large enough to contain the sqlm_stmt statically-sized elements, as well as a statement of the largest size to be extracted; that is, `SQL_MAX_STMT_SZ` plus sizeof(sqlm_stmt).

iSourceType
: Input. The type of source stream. Valid values are `SQLM_STREAM_SNAPSHOT` (snapshot stream), or `SQLM_STREAM_EVMON` (event monitor stream).

Usage Notes

Following is a list of supported convertable data elements:

```
Snapshot Variable Datastream Type            Structure
---------------------------------            ---------
SQLM_ELM_APPL                                sqlm_appl
SQLM_ELM_APPL_INFO                           sqlm_applinfo
SQLM_ELM_DB2                                 sqlm_db2
SQLM_ELM_FCM                                 sqlm_fcm
SQLM_ELM_FCM_NODE                            sqlm_fcm_node
SQLM_ELM_DBASE                               sqlm_dbase
SQLM_ELM_TABLE_LIST                          sqlm_table_header
SQLM_ELM_TABLE                               sqlm_table
SQLM_ELM_DB_LOCK_LIST                        sqlm_dbase_lock
SQLM_ELM_APPL_LOCK_LIST                      sqlm_appl_lock
SQLM_ELM_LOCK                                sqlm_lock
SQLM_ELM_STMT                                sqlm_stmt
SQLM_ELM_SUBSECTION                          sqlm_subsectiion
SQLM_ELM_TABLESPACE_LIST                     sqlm_tablespace_header
```

```
SQLM_ELM_TABLESPACE                       sqlm_tablespace
SQLM_ELM_ROLLFORWARD                      sqlm_rollfwd_info
SQLM_ELM_BUFFERPOOL                       sqlm_bufferpool
SQLM_ELM_LOCK_WAIT                        sqlm_lockwait
SQLM_ELM_DCS_APPL                         sqlm_dcs_appl, sqlm_dcs_applid_info,
                                          sqlm_dcs_appl_snap_stats,
                                          sqlm_xid, sqlm_tpmon
SQLM_ELM_DCS_DBASE                        sqlm_dcs_dbase
SQLM_ELM_DCS_APPL_INFO                    sqlm_dcs_applid_info
SQLM_ELM_DCS_STMT                         sqlm_dcs_stmt
SQLM_ELM_COLLECTED                        sqlm_collected

Event Monitor Variable Datastream Type    Structure
--------------------------------------    ---------
SQLM_ELM_EVENT_DB                         sqlm_db_event
SQLM_ELM_EVENT_CONN                       sqlm_conn_event
SQLM_ELM_EVENT_TABLE                      sqlm_table_event
SQLM_ELM_EVENT_STMT                       sqlm_stmt_event
SQLM_ELM_EVENT_XACT                       sqlm_xaction_event
SQLM_ELM_EVENT_DEADLOCK                   sqlm_deadlock_event
SQLM_ELM_EVENT_DLCONN                     sqlm_dlconn_event
SQLM_ELM_EVENT_TABLESPACE                 sqlm_tablespace_event
SQLM_ELM_EVENT_DBHEADER                   sqlm_dbheader_event
SQLM_ELM_EVENT_START                      sqlm_evmon_start_event
SQLM_ELM_EVENT_CONNHEADER                 sqlm_connheader_event
SQLM_ELM_EVENT_OVERFLOW                   sqlm_overflow_event
SQLM_ELM_EVENT_BUFFERPOOL                 sqlm_bufferpool_event
SQLM_ELM_EVENT_SUBSECTION                 sqlm_subsection_event
SQLM_ELM_EVENT_LOG_HEADER                 sqlm_event_log_header
```

The *sqlm_rollfwd_ts_info* structure is not converted; it only contains a table space name that can be accessed directly from the stream. The *sqlm_agent* structure is also not converted; it only contains the *pid* of the agent, which can also be accessed directly from the stream.

db2eva - Event Analyzer

Starts the event analyzer, allowing the user to trace performance data produced by DB2 event monitors that have their data directed to files.

Authorization

None, unless connecting to the database and selecting from the catalogs (-evm, -db, and -conn); then one of the following is required:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Required Connection

None

Command Syntax

```
►►──db2eva──┬─-path──evmon-target──┬─────────────────────────────┬─┬──────────►◄
            │                      └─-conn─┬─────────────────────┤ │
            │                              └─-db──-database-alias─┘ │
            └─-evm──evmon-name──-db──database-alias─┬───────┬──────┘
                                                    └─-conn─┘
```

Command Parameters

-path evmon-target
: Specifies the directory containing the event monitor trace files.

-conn
: Requests that **db2eva** maintain a connection to the database specified with -db, or if -db is not used, then to the database specified in the event monitor trace header. Maintaining a connection allows the event analyzer to obtain information not contained in the trace files (for example, the text for static SQL). A statement event record contains the package creator, package, and section number; when -conn is specified, **db2eva** can retrieve the text from the database system catalog (sysibm.sysstmt).

-db database-alias
: Specifies the name of the database defined for the event monitor. If -path is specified, the database name in the event monitor trace header is overridden.

-evm evmon-name
: Specifies the name of the event monitor whose traces are to be analyzed.

Usage Notes

Although there is no required connection, **db2eva** will attempt to connect to the database if the `-conn`, or the `-evm` and the `-db` options are used. If the user can access the database and has the appropriate authorization, the SQL text for static statements can be displayed. Without the required access or authority, only the text for dynamic statements is available.

There are two methods for reading event monitor traces:

1. Specifying the directory where the trace files are located (using the `-path` option). This allows users to move trace files from a server and analyze them locally. This can be done even if the event monitor has been dropped.
2. Specifying the database and event monitor names allows automatic location of the trace files. The event analyzer connects to the database, and issues a `select target from sysibm.syseventmonitors` to locate the directory where the event monitor writes its trace files. The connection is then released, unless `-conn` was specified. This method cannot be used if the event monitor has been dropped.

Note: The event analyzer can be used to analyze the data produced by an active event monitor. However, event monitors buffer their data before writing it to disk; therefore, some information may be missing. Turn off the event monitor, thereby forcing it to flush its buffers.

db2evmon - Event Monitor Productivity Tool

Formats event monitor file and named pipe output, and writes it to standard output.

Note: This productivity tool is provided *as is*, without any warranty of any kind, including the warranties of merchantability and fitness for a particular purpose, which are expressly disclaimed.

Authorization

None, unless connecting to the database (`-evm`, `-db`,); then, one of the following is required:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

Required Connection

None

Command Syntax

```
►►──db2evmon──┬─┬──────────────────────────────────────────────┬─┬─────────────►◄
              │ └─-db──database-alias───-evm──event-monitor-name──┘ │
              └─-path──event-monitor-target─────────────────────────┘
```

Command Parameters

-db database-alias
: Specifies the database whose data is to be displayed. This parameter is case sensitive.

-evm event-monitor-name
: The one-part name of the event monitor. An ordinary or delimited SQL identifier. This parameter is case sensitive.

-path event-monitor-target
: Specifies the directory containing the event monitor trace files.

Usage Notes

If the data is being written to files, the tool formats the files for display using standard output. In this case, the monitor is turned on first, and any event data in the files is displayed by the tool. To view any data written to files after the tool has been run, reissue **db2evmon**.

If the data is being written to a pipe, the tool formats the output for display using standard output as events occur. In this case, the tool is started *before* the monitor is turned on.

db2GetSnapshot - Get Snapshot

Collects database manager monitor information and returns it to a user-allocated data buffer. The information returned represents a *snapshot* of the database manager operational status at the time the API was called.

Scope

This API returns information only for the node on which it is issued.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To obtain a snapshot from a remote instance (or a different local instance), it is necessary to first attach to that instance.

API Include File

db2ApiDf.h

C API Syntax

```
/* File: db2ApiDf.h */
/* API: Get Snapshot */
/* ... */
int db2GetSnapshot (
  unsigned char version,
  db2GetSnapshotData * data,
  struct sqlca * pSqlca);

typedef struct
{
  sqlma * piSqlmaData;
  sqlm_collected * poCollectedData;
  void * poBuffer;
  db2Uint32 iVersion;
  db2Uint32 iBufferSize;
  db2Uint32 iStoreResult
} db2GetSnapshotData;
/* ... */
```

Generic API Syntax

```
/* File: db2ApiDf.h */
/* API: Get Snapshot */
/* ... */
int db2GetSnapshot (
  unsigned char version,
  db2GetSnapshotData * data,
  struct sqlca * pSqlca);

typedef struct
{
  sqlma * piSqlmaData;
  sqlm_collected * poCollectedData;
  void * poBuffer;
  db2Uint32 iVersion;
  db2Uint32 iBufferSize;
  db2Uint32 iStoreResult
} db2GetSnapshotData;
/* ... */
```

API Parameters

version
Input. Specifies the version and release level of the structure passed in as the second parameter, *data*.

data Input/Output. A pointer to the *db2GetSnapshotData* structure.

pSqlca
Output. A pointer to the *sqlca* structure. For more information about this structure, see the *Administrative API Reference*.

piSqlmaData
: Input. Pointer to the user-allocated *sqlma* (monitor area) structure. This structure specifies the type(s) of data to be collected.

poCollectedData
: Output. A pointer to the *sqlm_collected* structure into which the database monitor delivers summary statistics and the number of each type of data structure returned in the buffer area.

  Note: This structure is only used for pre-Version 6 data streams. However, if a snapshot call is made to a back-level remote server, this structure must be passed in for results to be processed. It is therefore recommended that this parameter always be passed in.

poBuffer
: Output. Pointer to the user-defined data area into which the snapshot information will be returned. For information about interpreting the data returned in this buffer, see the *System Monitor Guide and Reference*.

iVersion
: Input. Version ID of the database monitor data to collect. The database monitor only returns data that was available for the requested version. Set this parameter to one of the following symbolic constants:

  - `SQLM_DBMON_VERSION1`
  - `SQLM_DBMON_VERSION2`
  - `SQLM_DBMON_VERSION5`
  - `SQLM_DBMON_VERSION5_2`
  - `SQLM_DBMON_VERSION6`

  If requesting data for a version higher than the current server, the database monitor only returns data for its level (see the *server_version* field in the ″collected″ portion of the datastream.

  Note: If `SQLM_DBMON_VERSION1` is specified as the version, the APIs cannot be run remotely.

iBufferSize
: Input. The length of the data buffer. Use “sqlmonsz - Estimate Size Required for db2GetSnapshot() Output Buffer” on page 358 to estimate the size of this buffer. If the buffer is not large enough, a warning is returned, along with the information that will fit in the assigned buffer. It may be necessary to resize the buffer and call the API again.

iStoreResult
: Input. An indicator set to `TRUE` or `FALSE`, depending on whether the snapshot results are to be stored at the DB2 server for viewing

through SQL. This parameter should only be set to `TRUE` when the snapshot is being taken over a database connection, and when one of the snapshot types in the *sqlma* is `SQLMA_DYNAMIC_SQL`.

Usage Notes

If an alias for a database residing at a different instance is specified, an error message is returned.

See Also

"db2ConvMonStream" on page 310

"sqlmon - Get/Update Monitor Switches" on page 355

"sqlmonsz - Estimate Size Required for db2GetSnapshot() Output Buffer" on page 358

"sqlmrset - Reset Monitor" on page 361.

DROP EVENT MONITOR Command and SQL

Removes an event monitor definition from the Database catalogs. Whenever an object is deleted, its description is deleted from the catalog and any packages that reference the object are invalidated.

Scope

This statement can be embedded in an application program or issued interactively. It is an executable statement that can be dynamically prepared.

Authorization

The authorization ID of the DROP statement when dropping an event monitor must have SYSADM or DBADM authority

Command Syntax

```
►►──DROP──EVENT──MONITOR──event-monitor-name───────────────────────────────►◄
```

Command Parameters

EVENT MONITOR *event-monitor-name*
Identifies the event monitor that is to be dropped. The *event-monitor-name* must identify an event monitor that is described in the catalog (SQLSTATE 42704).

If the identified event monitor is ON, an error (SQLSTATE 55034) is raised. Otherwise, the event monitor is deleted.

If there are event files in the target path of the event monitor when the event monitor is dropped, the event files are not deleted.

Sample Programs

An event monitor must be stopped or OFF before it can be deleted. Dropping an event monitor does not erase the target directory.

EVENT_MON_STATE

```
►►──EVENT_MON_STATE──(───string-expression──)──────────────────────────────►◄
```

The schema is SYSIBM.

The EVENT_MON_STATE function returns the current state of an event monitor.

The argument is a string expression with a resulting type of CHAR or VARCHAR and a value that is the name of an event monitor. If the named event monitor does not exist in the SYSCAT.EVENTMONITORS catalog table, SQLSTATE 42704 will be returned.

The result is an integer with one of the following values:

0	The event monitor is inactive.
1	The event monitor is active.

If the argument can be null, the result can be null; if the argument is null, the result is the null value.

Example:

- The following example selects all of the defined event monitors, and indicates whether each is active or inactive:

```
SELECT EVMONNAME,
   CASE
      WHEN EVENT_MON_STATE(EVMONNAME) = 0 THEN 'Inactive'
      WHEN EVENT_MON_STATE(EVMONNAME) = 1 THEN 'Active'
   END
FROM SYSCAT.EVENTMONITORS
```

FLUSH EVENT MONITOR

The FLUSH EVENT MONITOR statement writes current database monitor values for all active monitor types associated with event monitor *event-monitor-name* to the event monitor I/O target. Hence, at any time a partial event record is available for event monitors that have low record generation frequency (such as a database event monitor). Such records are noted in the event monitor log with a *partial record* identifier.

When an event monitor is flushed, its active internal buffers are written to the event monitor output object.

Scope

This statement can be embedded in an application program or issued interactively. It is an executable statement that can be dynamically prepared.

Authorization

The privileges held by the authorization ID must include either SYSADM or DBADM authority (SQLSTATE 42502).

Command Syntax

```
>>──FLUSH──EVENT──MONITOR──event-monitor-name──┬────────┬──────────────────────────><
                                               └─BUFFER─┘
```

Command Parameters

event-monitor-name
: Name of the event monitor. This is a one-part name. It is an SQL identifier.

BUFFER
: Indicates that the event monitor buffers are to be written out. If BUFFER is specified, then a partial record is not generated. Only the data already present in the event monitor buffers are written out.

Usage Notes

- Flushing out the event monitor will not cause the event monitor values to be reset. This means that the event monitor record that would have been generated if no flush was performed, will still be generated when the normal monitor event is triggered.

GET DATABASE MANAGER MONITOR SWITCHES

Displays the status of the database system monitor switches. Monitor switches instruct the database system manager to collect database activity information. Each application using the database system monitor interface has its own set of monitor switches (see “GET MONITOR SWITCHES” on page 326). A database manager-level switch is on when any of the monitoring applications has turned it on. This command is used to determine if the database system monitor is currently collecting data for any monitoring application.

Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:
- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To display the settings for a remote instance, or for a different local instance, it is necessary to first attach to that instance.

Command Syntax

```
►►──GET───┬─DATABASE MANAGER─┬───MONITOR SWITCHES──────────────────────────────────►◄
          ├─DB MANAGER───────┤
          └─DBM──────────────┘
```

Command Parameters

None

Examples

The following is sample output from GET DATABASE MANAGER MONITOR SWITCHES:

```
            DBM System Monitor Information Collected

Buffer Pool Activity Information (BUFFERPOOL) = ON   06-11-1997 10:11:01.738377
Lock Information                       (LOCK) = OFF
Sorting Information                    (SORT) = ON   06-11-1997 10:11:01.738400
SQL Statement Information         (STATEMENT) = OFF
Table Activity Information            (TABLE) = OFF
Unit of Work Information                (UOW) = ON   06-11-1997 10:11:01.738353
```

Usage Notes

The six recording switches (BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, and UOW) are off by default, but may be switched on using "UPDATE MONITOR SWITCHES" on page 364. If a particular switch is on, this command also displays the time stamp for when the switch was turned on.

See Also

"GET MONITOR SWITCHES" on page 326

"GET SNAPSHOT" on page 328

"RESET MONITOR" on page 349

"UPDATE MONITOR SWITCHES" on page 364.

GET MONITOR SWITCHES

Displays the status of the database system monitor switches for the current session. Monitor switches instruct the database system manager to collect database activity information. Each application using the database system monitor interface has its own set of monitor switches. This command displays them. To display the database manager-level switches, use “GET DATABASE MANAGER MONITOR SWITCHES” on page 324.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To display the settings for a remote instance, or for a different local instance, it is necessary to first attach to that instance.

Command Syntax

```
►►──GET MONITOR SWITCHES──────────────────────────────────────────────►◄
```

Command Parameters

None

Examples

The following is sample output from GET MONITOR SWITCHES:

```
            Monitor Recording Switches

Buffer Pool Activity Information  (BUFFERPOOL) = ON  02-20-1997 16:04:30.070073
Lock Information                        (LOCK) = OFF
Sorting Information                     (SORT) = OFF
SQL Statement Information          (STATEMENT) = ON  02-20-1997 16:04:30.070073
Table Activity Information             (TABLE) = OFF
Unit of Work Information                 (UOW) = ON  02-20-1997 16:04:30.070073
```

Usage Notes

The six recording switches (BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, and UOW) are off by default, but may be switched on using "UPDATE MONITOR SWITCHES" on page 364. If a particular switch is on, this command also displays the time stamp for when the switch was turned on.

See Also

"GET DATABASE MANAGER MONITOR SWITCHES" on page 324

"GET SNAPSHOT" on page 328

"RESET MONITOR" on page 349

"UPDATE MONITOR SWITCHES" on page 364.

GET SNAPSHOT

Collects database manager status information and returns it to a user-allocated data buffer. The information returned represents a *snapshot* of the database manager operational status at the time the command was issued.

Scope

This command can be invoked from any node in the `db2nodes.cfg` file. It acts only on that node or partition.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To obtain a snapshot of a remote instance, it is necessary to first attach to that instance.

Command Syntax

```
►►—GET SNAPSHOT FOR——————————————————————————————————————————————————►
```

```
►──┬──┬─DATABASE MANAGER─┬──────────────────────────────────────────┬──────►◄
   │  ├─DB MANAGER───────┤                                          │
   │  └─DBM──────────────┘                                          │
   ├─ALL─┬───────┬─DATABASES────────────────────────────────────────┤
   │     └─DCS───┘                                                  │
   ├─ALL─┬───────┬─APPLICATIONS─────────────────────────────────────┤
   │     └─DCS───┘                                                  │
   ├─ALL BUFFERPOOLS────────────────────────────────────────────────┤
   ├──┬─────┬─APPLICATION──────┬─APPLID──appl-id──────────┬─────────┤
   │  └─DCS─┘                  └─AGENTID──appl-handle─────┘         │
   ├─FCM FOR ALL NODES──────────────────────────────────────────────┤
   ├─LOCKS FOR APPLICATION──┬─APPLID──appl-id─────────┬─────────────┤
   │                        └─AGENTID──appl-handle────┘             │
   └──┬─ALL───────────────────────┬─ON──database-alias──┬───────────────┬─┘
      ├──┬─────┬──┬─DATABASE─┬────┤                     └─WRITE TO FILE─┘
      │  └─DCS─┘  └─DB───────┘    │
      ├──┬─────┬─APPLICATIONS─────┤
      │  └─DCS─┘                  │
      ├─TABLES────────────────────┤
      ├─TABLESPACES───────────────┤
      ├─LOCKS─────────────────────┤
      ├─BUFFERPOOLS───────────────┤
      └─DYNAMIC SQL───────────────┘
```

Note: The monitor switches must be turned on to get some statistics (see "UPDATE MONITOR SWITCHES" on page 364).

Command Parameters

DATABASE MANAGER
Provides statistics for the active database manager instance.

ALL DATABASES
Provides general statistics for all active databases on the current node.

ALL APPLICATIONS
Provides information about all active applications that are connected to a database on the current node.

ALL BUFFERPOOLS
Provides information about buffer pool activity for all active databases.

APPLICATION APPLID appl-id
Provides information only about the application whose ID is specified. To get a specific application ID, use "LIST APPLICATIONS" on page 344.

APPLICATION AGENTID appl-handle
Provides information only about the application whose application handle is specified. The application handle is a 32-bit number that

uniquely identifies an application that is currently running. Use "LIST APPLICATIONS" on page 344 to get a specific application handle.

FCM FOR ALL NODES
Provides FCM statistics for all nodes.

LOCKS FOR APPLICATION APPLID appl-id
Provides information about all locks held by the specified application, identified by application ID.

LOCKS FOR APPLICATION AGENTID appl-handle
Provides information about all locks held by the specified application, identified by application handle.

ALL ON database-alias
Provides general statistics and information about all applications, tables, table spaces, buffer pools, and locks for a specified database.

DATABASE ON database-alias
Provides general statistics for a specified database.

APPLICATIONS ON database-alias
Provides information about all applications connected to a specified database.

TABLES ON database-alias
Provides information about tables in a specified database. This will include only those tables that have been accessed since the TABLE recording switch was turned on.

TABLESPACES ON database-alias
Provides information about table spaces for a specified database.

LOCKS ON database-alias
Provides information about every lock held by each application connected to a specified database.

BUFFERPOOLS ON database-alias
Provides information about buffer pool activity for the specified database.

DYNAMIC SQL ON database-alias
Returns a point-in-time picture of the contents of the SQL statement cache for the database.

WRITE TO FILE
Only available for DYNAMIC SQL snapshots. Specifies that snapshot results are to be stored in a file at the server, as well as being passed back to the client. This command is valid only over a database connection. The snapshot data can then be queried through the table

function SYSFUN.SQLCACHE_SNAPSHOT over the same connection on which the call was made. For more information, see the *System Monitor Guide and Reference*.

DCS Depending on which clause it is specified, this keyword requests statistics about:

- A specific DCS application currently running on the DB2 Connect Gateway
- All DCS applications
- All DCS applications currently connected to a specific DCS database
- A specific DCS database
- All DCS databases.

Examples

In the following sample output listings, some of the information may not be available, depending on whether or not the appropriate database system monitor recording switch is turned on (see "UPDATE MONITOR SWITCHES" on page 364). If the information is unavailable, `Not Collected` appears in the output.

The following is typical output resulting from a request for database manager information:

```
            Database Manager Snapshot

Node type                                      = Database Server with local clients
Instance name                                  = smith
Number of nodes in DB2 instance                = 0
Database manager status                        = Active

Product name                                   =
Product identification                         =
Service level                                  =

Sort heap allocated                            = 0
Post threshold sorts                           = 0
Piped sorts requested                          = 0
Piped sorts accepted                           = 0

Start Database Manager timestamp               = 02-25-1999 13:26:53.126518
Last reset timestamp                           =
Snapshot timestamp                             = 02-25-1999 13:45:42.257720

Remote connections to db manager               = 0
Remote connections executing in db manager     = 0
Local connections                              = 1
Local connections executing in db manager      = 0
Active local databases                         = 1

High water mark for agents registered          = 3
High water mark for agents waiting for a token = 0
Agents registered                              = 3
Agents waiting for a token                     = 0
Idle agents                                    = 1
```

```
Committed private Memory (Bytes)              = 3670016

Buffer Pool Activity Information  (BUFFERPOOL) = ON  02-25-1999 13:32:14
Lock Information                        (LOCK) = ON  02-25-1999 13:32:40
Sorting Information                     (SORT) = ON  02-25-1999 13:32:40
SQL Statement Information          (STATEMENT) = ON  02-25-1999 13:32:14
Table Activity Information             (TABLE) = ON  02-25-1999 13:32:40
Unit of Work Information                 (UOW) = ON  02-25-1999 13:32:14

Agents assigned from pool                      = 2
Agents created from empty pool                 = 3
Agents stolen from another application         = 0
High water mark for coordinating agents        = 3
Max agents overflow                            = 0
Hash joins after heap threshold exceeded       = 0

Total number of gateway connections            = 0
Current number of gateway connections          = 0
Gateway connections waiting for host reply     = 0
Gateway connections waiting for client reply   = 0
Gateway inactive connection pool agents        = 0
Gateway connection pool agents stolen          = 0
```

The following is typical output resulting from a request for database information:

```
              Database Snapshot

Database name                              = SAMPLE
Database path                              = /home/smith/smith/NODE0000/SQL00001/
Input database alias                       =
Database status                            = Active
Catalog node number                        = 0
Catalog network node name                  =
Operating system running at database server= AIX
Location of the database                   = Local
First database connect timestamp           = 02-25-1999 13:31:33.886214
Last reset timestamp                       =
Last backup timestamp                      =
Snapshot timestamp                         = 02-25-1999 13:40:08.337902

High water mark for connections            = 1
Application connects                       = 1
Secondary connects total                   = 0
Applications connected currently           = 1
Appls. executing in db manager currently   = 0
Agents associated with applications        = 1
Maximum agents associated with applications= 1
Maximum coordinating agents                = 1

Locks held currently                       = 1
Lock waits                                 = 0
Time database waited on locks (ms)         = 0
Lock list memory in use (Bytes)            = 432
Deadlocks detected                         = 0
Lock escalations                           = 0
Exclusive lock escalations                 = 0
Agents currently waiting on locks          = 0
Lock Timeouts                              = 0

Total sort heap allocated                  = 0
Total sorts                                = 0
Total sort time (ms)                       = 0
Sort overflows                             = 0
Active sorts                               = 0
```

```
High water mark for database heap          = 316084

Buffer pool data logical reads             = 1
Buffer pool data physical reads            = 0
Asynchronous pool data page reads          = 0
Buffer pool data writes                    = 0
Asynchronous pool data page writes         = 0
Buffer pool index logical reads            = 0
Buffer pool index physical reads           = 0
Asynchronous pool index page reads         = 0
Buffer pool index writes                   = 0
Asynchronous pool index page writes        = 0
Total buffer pool read time (ms)           = 0
Total buffer pool write time (ms)          = 0
Total elapsed asynchronous read time       = 0
Total elapsed asynchronous write time      = 0
Asynchronous read requests                 = 0
LSN Gap cleaner triggers                   = 0
Dirty page steal cleaner triggers          = 0
Dirty page threshold cleaner triggers      = 0
Time waited for prefetch (ms)              = 0
Direct reads                               = 0
Direct writes                              = 0
Direct read requests                       = 0
Direct write requests                      = 0
Direct reads elapsed time (ms)             = 0
Direct write elapsed time (ms)             = 0
Database files closed                      = 0
Data pages copied to extended storage      = 0
Index pages copied to extended storage     = 0
Data pages copied from extended storage    = 0
Index pages copied from extended storage   = 0

Commit statements attempted                = 2
Rollback statements attempted              = 0
Dynamic statements attempted               = 10
Static statements attempted                = 2
Failed statement operations                = 0
Select SQL statements executed             = 2
Update/Insert/Delete statements executed   = 0
DDL statements executed                    = 0

Internal automatic rebinds                 = 0
Internal rows deleted                      = 0
Internal rows inserted                     = 0
Internal rows updated                      = 0
Internal commits                           = 1
Internal rollbacks                         = 0
Internal rollbacks due to deadlock         = 0

Rows deleted                               = 0
Rows inserted                              = 0
Rows updated                               = 0
Rows selected                              = 16

Binds/precompiles attempted                = 0

Log space available to the database (Bytes)= 0
Log space used by the database (Bytes)     = 0
Maximum secondary log space used (Bytes)   = 0
Maximum total log space used (Bytes)       = 0
Secondary logs allocated currently         = 0
Log pages read                             = 0
Log pages written                          = 0
Appl id holding the oldest transaction     = 0
```

```
Package cache lookups                     = 2
Package cache inserts                     = 1
Package cache overflows                   = 0
Package cache high water mark (Bytes)     = 108757
Application section lookups               = 10
Application section inserts               = 1

Catalog cache lookups                     = 1
Catalog cache inserts                     = 1
Catalog cache overflows                   = 0
Catalog cache heap full                   = 0

Number of hash joins                      = 0
Number of hash loops                      = 0
Number of hash join overflows             = 0
Number of small hash join overflows       = 0
```

The following is typical output resulting from a request for DCS database information:

```
             DCS Database Snapshot

DCS database name                         = DCSDB
Host database name                        = GILROY
First database connect timestamp          = 02-25-1999 17:00:05.003421
Most recent elapsed time to connect       = 0.001200
Most recent elapsed connection duration   = 3.443780
Host response time (sec.ms)               = 0.000320
Last reset timestamp                      =
Number of SQL statements attempted        = 12
Commit statements attempted               = 6
Rollback statements attempted             = 2
Failed statement operations               = 4
Total number of gateway connections       = 0
Current number of gateway connections     = 1
Gateway conn. waiting for host reply      = 0
Gateway conn. waiting for client reply    = 1
Gateway communication errors to host      = 0
Timestamp of last communication error     = None
High water mark for gateway connections   = 1
Rows selected                             = 0
Outbound bytes sent                       = 0
Outbound bytes received                   = 0
```

The following is typical output resulting from a request for application information (by specifying either an application ID, an application handle, all applications, or all applications on a database):

```
            Application Snapshot

Application handle                        = 3
Application status                        = UOW Waiting
Status change time                        = 02-25-1999 13:33:41.446676
Application code page                     = 819
Application country code                  = 1
DUOW correlation token                    = *LOCAL.smith.990225183133
Application name                          = db2bp
Application ID                            = *LOCAL.smith.990225183133
Sequence number                           = 0001
Connection request start timestamp        = 02-25-1999 13:31:33.886214
Connect request completion timestamp      = 02-25-1999 13:31:34.434114
Application idle time                     = 6 minutes and 42 seconds
Authorization ID                          = SMITH
```

```
Client login ID                          = smith
Configuration NNAME of client            =
Client database manager product ID       = SQL06000
Process ID of client application         = 27918
Platform of client application           = AIX
Communication protocol of client         = Local Client

Outbound communication address           =
Outbound communication protocol          = APPC
Inbound communication address            =

Database name                            = SAMPLE
Database path                            = /home/smith/smith/NODE0000/SQL00001/
Client database alias                    = sample
Input database alias                     =
Last reset timestamp                     =
Snapshot timestamp                       = 02-25-1999 13:40:23.773540
The highest authority level granted      =
        Direct DBADM authority
        Direct CREATETAB authority
        Direct BINDADD authority
        Direct CONNECT authority
        Direct CREATE_NOT_FENC authority
        Direct IMPLICIT_SCHEMA authority
        Indirect SYSADM authority
        Indirect CREATETAB authority
        Indirect BINDADD authority
        Indirect CONNECT authority
        Indirect IMPLICIT_SCHEMA authority
Coordinating node number                 = 0
Current node number                      = 0
Coordinator agent process or thread ID   = 26160
Agents stolen                            = 0
Agents waiting on locks                  = 0
Maximum associated agents                = 1
Priority at which application agents work  = 0
Priority type                            = Dynamic

Locks held by application                = 1
Lock waits since connect                 = 0
Time application waited on locks (ms)    = 0
Deadlocks detected                       = 0
Lock escalations                         = 0
Exclusive lock escalations               = 0
Number of Lock Timeouts since connected  = 0
Total time UOW waited on locks (ms)      = 0

Total sorts                              = 0
Total sort time (ms)                     = 0
Total sort overflows                     = 0

Data pages copied to extended storage    = 0
Index pages copied to extended storage   = 0
Data pages copied from extended storage  = 0
Index pages copied from extended storage = 0
Buffer pool data logical reads           = 1
Buffer pool data physical reads          = 0
Buffer pool data writes                  = 0
Buffer pool index logical reads          = 0
Buffer pool index physical reads         = 0
Buffer pool index writes                 = 0
Total buffer pool read time (ms)         = 0
Total buffer pool write time (ms)        = 0
Time waited for prefetch (ms)            = 0
Direct reads                             = 0
Direct writes                            = 0
Direct read requests                       = 0
Direct write requests                      = 0
Direct reads elapsed time (ms)             = 0
Direct write elapsed time (ms)             = 0

Number of SQL requests since last commit   = 5
Commit statements                          = 2
Rollback statements                        = 0
Dynamic SQL statements attempted           = 10
Static SQL statements attempted            = 2
Failed statement operations                = 0
Select SQL statements executed             = 2
Update/Insert/Delete statements executed   = 0
DDL statements executed                    = 0
Internal automatic rebinds                 = 0
Internal rows deleted                      = 0
Internal rows inserted                     = 0
Internal rows updated                      = 0
Internal commits                           = 1
Internal rollbacks                         = 0
Internal rollbacks due to deadlock         = 0
Binds/precompiles attempted                = 0
Rows deleted                               = 0
Rows inserted                              = 0
Rows updated                               = 0
Rows selected                              = 16
Rows read                                  = 25
Rows written                               = 0

UOW log space used (Bytes)                 = 0
Previous UOW completion timestamp          = 02-25-1999 13:31:34.434114
Elapsed time of last completed uow (sec.ms)= 0.919533380
UOW start timestamp                        = 02-25-1999 13:33:41.392167
UOW stop timestamp                         =
UOW completion status                      =
Open remote cursors                        = 0
Open remote cursors with blocking          = 0
Rejected Block Remote Cursor requests      = 0
Accepted Block Remote Cursor requests      = 2
Open local cursors                         = 0
Open local cursors with blocking           = 0

Total User CPU Time used by agent (s)      = 0.100000
Total System CPU Time used by agent (s)    = 0.020000
Package cache lookups                      = 2
Package cache inserts                      = 1
Application section lookups                = 10
Application section inserts                = 1
Catalog cache lookups                      = 1
Catalog cache inserts                      = 1
Catalog cache overflows                    = 0
Catalog cache heap full                    = 0

Most recent operation                      = Select
Most recent operation start timestamp      = 02-25-1999 13:33:41.394260
Most recent operation stop timestamp       = 02-25-1999 13:33:41.446740
Agents associated with the application     = 1

Number of hash joins                       = 0
Number of hash loops                       = 0
Number of hash join overflows              = 0
Number of small hash join overflows        = 0

Statement type                             = Dynamic SQL Statement
Statement                                  = Select
Section number                             = 201
```

```
Application creator                       = NULLID
Package name                              = SQLC28A4
Cursor name                               = SQLCUR201
Statement node number                     = 0
Statement start timestamp                 = 02-25-1999 13:33:41.394260
Statement stop timestamp                  = 02-25-1999 13:33:41.446740
Elapsed time of last completed stmt(sec.ms)= 0.000000
Total user CPU time                       = 0.000000
Total system CPU time                     = 0.000000
SQL compiler cost estimate in timerons    = 30
SQL compiler cardinality estimate         = 47
Degree of parallelism requested           = 1
Number of agents working on statement     = 1
Number of subagents created for statement = 1
Statement sorts                           = 0
Total sort time                           = 0
Sort overflows                            = 0
Rows read                                 = 8
Rows written                              = 0
Rows deleted                              = 0
Rows updated                              = 0
Rows inserted                             = 0
Rows fetched                              = 0
Number of subsections                     = 0
Dynamic SQL statement text:
select * from org
```

The following is typical output resulting from a request for DCS application information (by specifying either a DCS application ID, a DCS application handle, all DCS applications, or all DCS applications on a database):

```
           DCS Application Snapshot
Client application ID                     = 09151251.04D6.980521202839
  Sequence number                         = 0001
  Authorization ID                        = NEWTON
  Application name                        = db2bp
  Application handle                      = 0
  Application status                      = waiting for request
  Status change time                      = 05-21-1998 16:35:27.670354
  Client DB alias                         = MVSDB
  Client node                             = antman
  Client release level                    = SQL05020
  Client platform                         = AIX
  Client protocol                         = TCP/IP
  Client codepage                         = 819
  Process ID of client application        = 35754
  Client login ID                         = user1
  Host application ID                     = G9151251.G4D7.980521202840
  Sequence number                         = 0000
  Host DB name                            = GILROY
  Host release level                      = DSN05011
  Host CCSID                              = 500
Outbound communication address            = 9.21.21.92 5021
Outbound communication protocol           = TCP/IP
Inbound communication address             = 9.31.12.34 334
First database connect timestamp          = 05-21-1998 16:28:39.517919
Time spent on gateway processing          = 0.334215
Last reset timestamp                      =
Rows selected                             = 0
Number of SQL statements attempted        = 2
Failed statement operations               = 0
Commit statements                         = 1
Rollback statements                       = 0
Inbound bytes received                    = 392
```

```
Outbound bytes sent                       = 136
Outbound bytes received                   = 178
Inbound bytes sent                        = 190
Number of open cursors                    = 0
Application idle time                     = 53 seconds
UOW completion status                     = Committed - Commit Statement
Previous UOW completion timestamp         =
UOW start timestamp                       = 05-21-1998 16:35:27.252375
UOW stop timestamp                        = 05-21-1998 16:35:27.670290
Inbound bytes received for UOW            = 180
Outbound bytes sent for UOW               = 136
Outbound bytes received for UOW           = 178
Inbound bytes sent for UOW                = 190
Most recent operation                     = Static Commit
Most recent operation start timestamp     = 05-21-1998 16:35:27.284183
Most recent operation stop timestamp      = 05-21-1998 16:35:27.670290
Statement                                 = Static Commit
Section number                            = 0
Application creator                       = NULLID
Package name                              = SQLC28A0
SQL compiler cost estimate in timerons    = 0
SQL compiler cardinality estimate         = 0
Statement start timestamp                 = 05-21-1998 16:35:27.284183
Statement stop timestamp                  = 05-21-1998 16:35:27.670290
Rows fetched                              = 0
Time spent on gateway processing          = 0.333740
Inbound bytes received for statement      = 0
Outbound bytes sent for statement         = 10
Outbound bytes received for statement     = 54
Inbound bytes sent for statement          = 0
```

The following is typical output resulting from a request for buffer pool information:

```
             Bufferpool Snapshot
Bufferpool name                           = IBMDEFAULTBP
Database name                             = SAMPLE
Database path                             = /home/user1/user1/NODE0000/SQL00011/
Input database alias                      = SAMPLE
Buffer pool data logical reads            = 32
Buffer pool data physical reads           = 13
Buffer pool data writes                   = 0
Buffer pool index logical reads           = 55
Buffer pool index physical reads          = 23
Total buffer pool read time (ms)          = 364
Total buffer pool write time (ms)         = 0
Database files closed                     = 0
Asynchronous pool data page reads         = 0
Asynchronous pool data page writes        = 0
Buffer pool index writes                  = 0
Asynchronous pool index page reads        = 0
Asynchronous pool index page writes       = 0
Total elapsed asynchronous read time      = 0
Total elapsed asynchronous write time     = 0
Asynchronous read requests                = 0
Direct reads                              = 34
Direct writes                             = 0
Direct read requests                      = 4
Direct write requests                     = 0
Direct reads elapsed time (ms)            = 1
Direct write elapsed time (ms)            = 0
Data pages copied to extended storage     = 0
Index pages copied to extended storage    = 0
Data pages copied from extended storage   = 0
Index pages copied from extended storage  = 0
```

The following is typical output resulting from a request for table space information:

```
             Tablespace Snapshot
First database connect timestamp           = 04-04-1997 14:29:55.197659
Last reset timestamp                       =
Snapshot timestamp                         = 04-04-1997 14:32:14.151875
Database name                              = SAMPLE
Database path                              = /home/user1/user1/NODE0000/SQL00011/
Input database alias                       = SAMPLE
Number of accessed tablespaces             = 3
Tablespace name                            = SYSCATSPACE
  Data pages copied to extended storage    = 0
  Index pages copied to extended storage   = 0
  Data pages copied from extended storage  = 0
  Index pages copied from extended storage = 0
  Buffer pool data logical reads           = 26
  Buffer pool data physical reads          = 11
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 55
  Buffer pool index physical reads         = 23
  Asynchronous pool index page reads       = 0
  Buffer pool index writes                 = 0
  Asynchronous pool index page writes      = 0
  Total buffer pool read time (ms)         = 342
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
  Asynchronous read requests               = 0
  Direct reads                             = 34
  Direct writes                            = 0
  Direct read requests                     = 4
  Direct write requests                    = 0
  Direct reads elapsed time (ms)           = 1
  Direct write elapsed time (ms)           = 0
  Number of files closed                   = 0
Tablespace name                            = TEMPSPACE1
  Data pages copied to extended storage    = 0
  Index pages copied to extended storage   = 0
  Data pages copied from extended storage  = 0
  Index pages copied from extended storage = 0
  Buffer pool data logical reads           = 0
  Buffer pool data physical reads          = 0
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 0
  Buffer pool index physical reads         = 0
  Asynchronous pool index page reads       = 0
  Buffer pool index writes                 = 0
  Asynchronous pool index page writes      = 0
  Total buffer pool read time (ms)         = 0
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
  Asynchronous read requests               = 0
  Direct reads                             = 0
  Direct writes                            = 0
  Direct read requests                     = 0
  Direct write requests                    = 0
  Direct reads elapsed time (ms)           = 0
  Direct write elapsed time (ms)           = 0
  Number of files closed                   = 0
Tablespace name                            = USERSPACE1
```

```
 Data pages copied to extended storage    = 0
 Index pages copied to extended storage   = 0
 Data pages copied from extended storage  = 0
 Index pages copied from extended storage = 0
 Buffer pool data logical reads           = 6
 Buffer pool data physical reads          = 2
 Asynchronous pool data page reads        = 0
 Buffer pool data writes                  = 0
 Asynchronous pool data page writes       = 0
 Buffer pool index logical reads          = 0
 Buffer pool index physical reads         = 0
 Asynchronous pool index page reads       = 0
 Buffer pool index writes                 = 0
 Asynchronous pool index page writes      = 0
 Total buffer pool read time (ms)         = 22
 Total buffer pool write time (ms)        = 0
 Total elapsed asynchronous read time     = 0
 Total elapsed asynchronous write time    = 0
 Asynchronous read requests               = 0
 Direct reads                             = 0
 Direct writes                            = 0
 Direct read requests                     = 0
 Direct write requests                    = 0
 Direct reads elapsed time (ms)           = 0
 Direct write elapsed time (ms)           = 0
 Number of files closed                   = 0
```

The following is typical output resulting from a request for dynamic SQL information:

```
     Dynamic SQL Snapshot Result

Database name                      = SAMPLE

Database path                      = /home/smith/smith/NODE0000/SQL00001/

Number of executions               = 2
Number of compilations             = 1
Worst preparation time (ms)        = 126
Best preparation time (ms)         = 126
Rows deleted                       = 0
Rows inserted                      = 0
Rows read                          = 24
Rows updated                       = 0
Rows written                       = 0
Statement sorts                    = 0
Total execution time (sec.ms)      = 0.060226
Total system cpu time (sec.ms)     = 0
Total user cpu time (sec.ms)       = 0
Statement text                     = select * from org
```

Usage Notes

To obtain a snapshot from a remote instance (or a different local instance), it is necessary to first attach to that instance. If an alias for a database residing at a different instance is specified, an error message is returned.

To obtain some statistics, it is necessary that the database system monitor switches are turned on.

No data is returned following a request for table information if any of the following is true:

- The TABLE recording switch is turned off.
- No tables have been accessed since the switch was turned on.
- No tables have been accessed since the last RESET MONITOR command was issued.

See Also

"GET MONITOR SWITCHES" on page 326

"LIST APPLICATIONS" on page 344

"RESET MONITOR" on page 349.

LIST ACTIVE DATABASES

Displays a subset of the information listed by the GET SNAPSHOT FOR ALL DATABASES command (see “GET SNAPSHOT” on page 328). For each active database, this command displays the following:

- Database name
- Number of applications currently connected to the database
- Database path.

Scope

This command can be issued from any node that is listed in `$HOME/sqllib/db2nodes.cfg`. It returns the same information from any of these nodes.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Command Syntax

```
►►──LIST ACTIVE DATABASES────────────────────────────────────────────────────►◄
```

Command Parameters

None

Examples

Following is sample output from the LIST ACTIVE DATABASES command:

```
                           Active Databases

Database name                           = TEST
Applications connected currently        = 0
Database path                           = /home/smith/smith/NODE0000/SQL00002/

Database name                           = SAMPLE
Applications connected currently        = 1
Database path                           = /home/smith/smith/NODE0000/SQL00001/
```

See Also

"GET SNAPSHOT" on page 328.

LIST APPLICATIONS

Displays to standard output the application program name, authorization ID (user name), application handle, application ID, and database name of all active database applications. This command can also optionally display an application's sequence number, status, status change time, and database path.

Scope

This command only returns information for the node on which it is issued.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance. To list applications for a remote instance, it is necessary to first attach to that instance.

Command Syntax

```
►►──LIST APPLICATIONS──┬──────────────────────────────────┬──┬─────────────┬──────►◄
                       └─FOR──┬─DATABASE─┬──database-alias─┘  └─SHOW DETAIL─┘
                              └─DB───────┘
```

Command Parameters

FOR DATABASE database-alias
Information for each application that is connected to the specified database is to be displayed. Database name information is not displayed. If this option is not specified, the command displays the information for each application that is currently connected to any database at the node to which the user is currently attached.

The default application information is comprised of the following:

- Authorization ID
- Application program name
- Application handle
- Application ID
- Database name.

SHOW DETAIL

Output will include the following additional information:

- Sequence #
- Application status
- Status change time
- Database path.

Note: If this option is specified, it is recommended that the output be redirected to a file, and that the report be viewed with the help of an editor. The output lines may wrap around when displayed on the screen.

Examples

The following is sample output from LIST APPLICATIONS:

```
Auth Id  Application    Appl.      Application Id                 DB       # of
         Name           Handle                                    Name    Agents
-------- -------------- ---------- ------------------------------ -------- -----
smith    db2bp_32       12         *LOCAL.smith.970220191502      TEST     1
smith    db2bp_32       11         *LOCAL.smith.970220191453      SAMPLE   1
```

Note: For more information about these fields, see the *System Monitor Guide and Reference*.

Usage Notes

The database administrator can use the output from this command as an aid to problem determination. In addition, this information is required if the database administrator wants to use “GET SNAPSHOT” on page 328 .

To list applications at a remote instance (or a different local instance), it is necessary to first attach to that instance. If FOR DATABASE is specified when an attachment exists, and the database resides at an instance which differs from the current attachment, the command will fail.

LIST DCS APPLICATIONS

Displays to standard output information about applications that are connected to host databases via DB2 Connect Enterprise Edition.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance. To list the DCS applications at a remote instance, it is necessary to first attach to that instance.

Command Syntax

```
►►──LIST DCS APPLICATIONS──┬──────────────┬──────────────────────────────►◄
                           ├─SHOW DETAIL─┤
                           └─EXTENDED────┘
```

Command Parameters

LIST DCS APPLICATIONS
The default application information includes:
- Host authorization ID (*username*)
- Application program name
- Application handle
- Outbound application ID (*luwid*).

SHOW DETAIL
Specifies that output include the following additional information:
- Client application ID
- Client sequence number
- Client database alias
- Client node name (*nname*)
- Client release level
- Client code page
- Outbound sequence number
- Host database name
- Host release level.

EXTENDED
Generates an extended report. This report includes all of the fields that are listed when the SHOW DETAIL option is specified, plus the following additional fields:
- DCS application status
- Status change time
- Client platform
- Client protocol
- Client code page
- Process ID of the client application
- Host coded character set ID (CCSID).

Examples

The following is sample output from LIST DCS APPLICATIONS:

```
   Auth Id  Application Name     Appl.      Outbound Application Id
                                 Handle
   -------- -------------------- ---------- --------------------------------
   DDCSUS1  db2bp_s              2          0915155C.139D.971205184245
```

The following is sample output from LIST DCS APPLICATIONS EXTENDED:

```
               List of DCS Applications - Extended Report

Client application ID                      = 09151251.0AD1.980529194106
  Sequence number                          = 0001
  Authorization ID                         = SMITH
  Application name                         = db2bp
  Application handle                       = 0
  Application status                       = waiting for reply
  Status change time                       = Not Collected
  Client DB alias                          = MVSDB
  Client node                              = antman
  Client release level                     = SQL05020
  Client platform                          = AIX
  Client protocol                          = TCP/IP
  Client codepage                          = 819
  Process ID of client application         = 38340
  Client login ID                          = user1
  Host application ID                      = G9151251.GAD2.980529194108
  Sequence number                          = 0000
  Host DB name                             = GILROY
  Host release level                       = DSN05011
  Host CCSID                               = 500
```

Notes:
1. The application status field contains one of the following values:

connect pending - outbound
Denotes that the request to connect to a host database has been issued, and that DB2 Connect is waiting for the connection to be established.

waiting for request
Denotes that the connection to the host database has been established, and that DB2 Connect is waiting for an SQL statement from the client application.

waiting for reply
Denotes that the SQL statement has been sent to the host database.

2. The status change time is shown only if the System Monitor UOW switch was turned on during processing. Otherwise, `Not Collected` is shown.
3. For more information about these fields, see the *System Monitor Guide and Reference*.

Usage Notes

The database administrator can use this command to match client application connections *to* the gateway with corresponding host connections *from* the gateway.

The database administrator can also use agent ID information to force specified applications off a DB2 Connect server.

RESET MONITOR

Resets the internal database system monitor data areas of a specified database, or of all active databases, to zero. The internal database system monitor data areas include the data areas for all applications connected to the database, as well as the data areas for the database itself.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To reset the monitor switches for a remote instance (or a different local instance), it is necessary to first attach to that instance.

Command Syntax

```
►►──RESET MONITOR──┬─ALL──┬─────┬──────────────────────────────────────┬──────────────►◄
                   │      └─DCS─┘                                      │
                   └─FOR──┬─────┬──┬─DATABASE─┬──database-alias────────┘
                          └─DCS─┘  └─DB───────┘
```

Command Parameters

ALL
: This option indicates that the internal counters should be reset for all databases.

FOR DATABASE database-alias
: This option indicates that only the database with alias *database-alias* should have its internal counters reset.

DCS
: Depending on which clause it is specified, this keyword resets the internal counters of:
  - All DCS databases
  - A specific DCS database.

Usage Notes

Each process (attachment) has its own private view of the monitor data. If one user resets, or turns off a monitor switch, other users are not affected. Change the setting of the monitor switch configuration parameters to make global changes to the monitor switches .

If ALL is specified, some database manager information is also reset to maintain consistency of the returned data, and some node-level counters are reset.

See Also

"GET SNAPSHOT" on page 328

"GET MONITOR SWITCHES" on page 326.

SET EVENT MONITOR STATE

The SET EVENT MONITOR STATE statement activates or deactivates an event monitor. The current state of an event monitor (active or inactive) is determined by using the EVENT_MON_STATE built-in function. The SET EVENT MONITOR STATE statement is not under transaction control.

Scope

This statement can be embedded in an application program or issued through the use of dynamic SQL statements. It is an executable statement that can be dynamically prepared. However, if the bind option DYNAMICRULES BIND applies, the statement cannot be dynamically prepared (SQLSTATE 42509).

Authorization

The authorization ID of the statement most hold either SYSADM or DBADM authority (SQLSTATE 42815).

Command Syntax

```
                                                    ┌─=─┐
►►──SET──EVENT──MONITOR──event-monitor-name──STATE──┴───┴──┬─0─────────────┬──────────►◄
                                                           ├─1─────────────┤
                                                           └─host-variable─┘
```

Command Parameters

event-monitor-name
Identifies the event monitor to activate or deactivate. The name must identify an event monitor that exists in the catalog (SQLSTATE 42704).

new-state
new-state can be specified either as an integer constant or as the name of a host variable that will contain the appropriate value at run time. The following may be specified:

0	Indicates that the specified event monitor should be deactivated.
1	Indicates that the specified event monitor should be activated. The event monitor should not already be active; otherwise a warning (SQLSTATE 01598) is issued.
host-variable	The data type is INTEGER. The value specified must be 0 or 1 (SQLSTATE 42815). If *host-variable* has an associated indicator variable, the value of that indicator variable must not indicate a null value (SQLSTATE 42815).

Sample Programs

- Although an unlimited number of event monitors may be defined, there is a limit of 32 event monitors that can be simultaneously active (SQLSTATE 54030).
- In order to activate an event monitor, the transaction in which the event monitor was created must have been committed (SQLSTATE 55033). This rule prevents (in one unit of work) creating an event monitor, activating the monitor, then rolling back the transaction.
- If the number or size of the event monitor files exceeds the values specified for MAXFILES or MAXFILESIZE on the CREATE EVENT MONITOR statement, an error (SQLSTATE 54031) is raised.
- If the target path of the event monitor (that was specified on the CREATE EVENT MONITOR statement) is already in use by another event monitor, an error (SQLSTATE 51026) is raised.

Usage Notes

- Activating an event monitor performs a reset of any counters associated with it.

The following example activates an event monitor called SMITHPAY.

```
SET EVENT MONITOR SMITHPAY STATE = 1
```

SQLCACHE_SNAPSHOT

```
►►──SQLCACHE_SNAPSHOT──────────────────────────────────────────────────►◄
```

The schema is SYSFUN.

The SQLCACHE_SNAPSHOT returns the the results of a snapshot of the db2 dynamic sql statement cache, when a snapshot is taken with iStoreResult set to true.

The results of the snapshot are returned to the buffer and written to the file *applid.sql* in the *tmp* subdirectory of the instance's *sqllib* subdirectory on the DB2 server. Where *applid* is the application ID of the user making the snapshot request. You can then access the snapshot data using the table function SQLCACHE_SNAPSHOT.

You can select specific columns by referencing data elements by their snapshot's name. For example:

```
 1) get snapshot for dynamic sql on foo write to file

 2) select table_name.db_name, table_name.num_executions from
    table(sysfun.SQLCACHE_SNAPSHOT()) table_name where
    table_name.commit_sql_stmts > 100
```

Where *table_name* is an arbitrary valid SQL identifier. The table function in this example is called *SQLCACHE_SNAPSHOT()*. This corresponds to a table function for returning information from a get snapshot for dynamic SQL.

For time and time and timestamp elements, the identifier names are *idname*_s and *idname*_ms. For example, for the data element total_exec_time, the identifiers would be total_exec_time_s and total_exec_time_ms.

The file containing the results of a dynamic SQL snapshot will be overwritten by the next dynamic SQL snapshot request. As well, any snapshot data files generated during the database connection are erased when the application disconnects.

The function does not take any arguments. A snapshot of the statement cache can only be taken over a database connection. If write to file is attempted over an instance attachement, the request will be rejected.

Table 4. Column names and data types of the table returned by SQLCACHE_SNAPSHOT table function

Column name	Data type
NUM_EXECUTIONS	INTEGER
NUM_COMPILATIONS	INTEGER
PREP_TIME_WORST	INTEGER
PREP_TIME_BEST	INTEGER
INT_ROWS_DELETED	INTEGER
INT_ROWS_INSERTED	INTEGER
ROWS_READ	INTEGER
INT_ROWS_UPDATED	INTEGER
ROWS_WRITE	INTEGER
STMT_SORTS	INTEGER
TOTAL_EXEC_TIME_S	INTEGER
TOTAL_EXEC_TIME_MS	INTEGER
TOT_U_CPU_TIME_S	INTEGER
TOT_U_CPU_TIME_MS	INTEGER
TOT_S_CPU_TIME_S	INTEGER
TOT_S_CPU_TIME_MS	INTEGER
DB_NAME	VARCHAR(8)
STMT_TEXT	CLOB(64K)

sqlmon - Get/Update Monitor Switches

Selectively turns on or off switches for groups of monitor data to be collected by the database manager. Returns the current state of these switches for the application issuing the call.

Scope

This API only returns information for the node on which it is executed.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To display the settings for a remote instance, or for a different local instance, it is necessary to first attach to that instance.

API Include File

sqlmon.h

C API Syntax

```
/* File: sqlmon.h */
/* API: Get/Update Monitor Switches */
/* ... */
int SQL_API_FN
  sqlmon (
    unsigned long        version,
    _SQLOLDCHAR          *reserved,
    sqlm_recording_group  group_states[],
    struct sqlca         *sqlca);
/* ... */
```

Generic API Syntax

```
/* File: sqlmon.h */
/* API: Get/Update Monitor Switches */
/* ... */
int SQL_API_FN
  sqlgmon (
    unsigned long         reserved_lgth,
    struct   sqlca        *sqlca,
    sqlm_recording_group  group_states[],
    _SQLOLDCHAR           *reserved,
    unsigned long         version);
/* ... */
```

API Parameters

reserved_lgth
Reserved for future use. Users should set this value to zero.

sqlca
Output. A pointer to the *sqlca* structure.

group_states
Input/Output. Pointer to an array of size SQLM_NUM_GROUPS (6). If the array size is less than six, an error message is returned. The user determines which element of the array corresponds to which switch by indexing it to the following symbolic statements (defined in `sqlmon.h`):

- SQLM_UOW_SW
- SQLM_STATEMENT_SW
- SQLM_TABLE_SW
- SQLM_BUFFER_POOL_SW
- SQLM_LOCK_SW
- SQLM_SORT_SW.

The array contains the following elements:

- An *input_state* element set to one of the following (defined in `sqlmon.h`):

  SQLM_ON
  Turns information group on.

  SQLM_OFF
  Turns information group off.

  SQLM_HOLD
  Leaves information group in its current state.

- An *output_state* element, containing current state information about the information group being monitored, is returned. `SQLM_ON` and `SQLM_OFF` indicate the state.

- A *start_time* element, indicating the time that the monitored group was turned on, is returned. If monitoring of this group is turned off, the time stamp is zero.

reserved
Reserved for future use. Users should set this value to NULL.

version
Input. Version ID of the database monitor data to collect. The database monitor only returns data that was available for the requested version. Set this parameter to one of the following symbolic constants:

- `SQLM_DBMON_VERSION1`
- `SQLM_DBMON_VERSION2`
- `SQLM_DBMON_VERSION5`
- `SQLM_DBMON_VERSION5_2`
- `SQLM_DBMON_VERSION6`

If requesting data for a version higher than the current server, the database monitor only returns data for its level (see the *server_version* field in the ″collected″ portion of the datastream.

Note: If `SQLM_DBMON_VERSION1` is specified as the version, the APIs cannot be run remotely.

Sample Programs

C \sqllib\samples\c\db2mon.c

Usage Notes

To obtain the status of the switches at the database manager level, call “db2GetSnapshot - Get Snapshot” on page 317, specifying `SQMA_DB2` for *OBJ_TYPE* (get snapshot for database manager).

See Also

“db2GetSnapshot - Get Snapshot” on page 317

“sqlmonsz - Estimate Size Required for db2GetSnapshot() Output Buffer” on page 358

“sqlmrset - Reset Monitor” on page 361.

sqlmonsz - Estimate Size Required for db2GetSnapshot() Output Buffer

Estimates the buffer size needed by "db2GetSnapshot - Get Snapshot" on page 317.

Scope

This API only affects the instance to which the calling application is attached.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To obtain information from a remote instance (or a different local instance), it is necessary to first attach to that instance. If an attachment does not exist, an implicit instance attachment is made to the node specified by the **DB2INSTANCE** environment variable.

API Include File

sqlmon.h

C API Syntax

```
/* File: sqlmon.h */
/* API: Estimate Size Required for sqlmonss() Output Buffer */
/* ... */
int SQL_API_FN
  sqlmonsz (
    unsigned long  version,
    _SQLOLDCHAR    *reserved,
    sqlma          *sqlma_ptr,
    unsigned long  *buff_size,
    struct sqlca   *sqlca);
/* ... */
```

Generic API Syntax

```
/* File: sqlmon.h */
/* API: Estimate Size Required for sqlmonss() Output Buffer */
/* ... */
int SQL_API_FN
  sqlgmnsz (
    unsigned long   reserved_lgth,
    struct   sqlca  *sqlca,
    unsigned long   *buff_size,
    sqlma           *sqlma_ptr,
    _SQLOLDCHAR     *reserved,
    unsigned long   version);
/* ... */
```

API Parameters

reserved_lgth
: Reserved for future use. This value should be set to zero.

sqlca
: Output. A pointer to the *sqlca* structure.

buff_size
: Output. A pointer to the returned estimated buffer size needed by the GET SNAPSHOT API.

sqlma_ptr
: Input. Pointer to the user-allocated *sqlma* (monitor area) structure. This structure specifies the type(s) of snapshot data to be collected, and can be reused as input to "db2GetSnapshot - Get Snapshot" on page 317.

reserved
: Reserved for future use. Must be set to NULL.

version
: Input. Version ID of the database monitor data to collect. The database monitor only returns data that was available for the requested version. Set this parameter to one of the following symbolic constants:

- `SQLM_DBMON_VERSION1`
- `SQLM_DBMON_VERSION2`
- `SQLM_DBMON_VERSION5`
- `SQLM_DBMON_VERSION5_2`
- `SQLM_DBMON_VERSION6`

Note: If `SQLM_DBMON_VERSION1` is specified as the version, the APIs cannot be run remotely.

Sample Programs

C \sqllib\samples\c\db2mon.c

Usage Notes

This function generates a significant amount of overhead. Allocating and freeing memory dynamically for each **db2GetSnapshot** call is also expensive. If calling **db2GetSnapshot** repeatedly, for example, when sampling data over a period of time, it may be preferable to allocate a buffer of fixed size, rather than call **sqlmonsz**.

If the database system monitor finds no active databases or applications, it may return a buffer size of zero (if, for example, lock information related to a database that is not active is requested). Verify that the estimated buffer size returned by this API is non-zero before calling "db2GetSnapshot - Get Snapshot" on page 317. If an error is returned by **db2GetSnapshot** because of insufficient buffer space to hold the output, call this API again to determine the new size requirements.

See Also

"sqlmon - Get/Update Monitor Switches" on page 355

"db2GetSnapshot - Get Snapshot" on page 317

"sqlmrset - Reset Monitor" on page 361.

sqlmrset - Reset Monitor

Resets the database system monitor data of a specified database, or of all active databases, for the application issuing the call.

Scope

This API only affects the node on which it is issued.

Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To reset the monitor switches for a remote instance (or a different local instance), it is necessary to first attach to that instance.

API Include File

sqlmon.h

C API Syntax

```
/* File: sqlmon.h */
/* API: Reset Monitor */
/* ... */
int SQL_API_FN
  sqlmrset (
    unsigned long  version,
    _SQLOLDCHAR    *reserved,
    unsigned long  reset_all,
    _SQLOLDCHAR    *db_alias,
    struct sqlca   *sqlca);
/* ... */
```

Generic API Syntax

```
/* File: sqlmon.h */
/* API: Reset Monitor */
/* ... */
int SQL_API_FN
  sqlgmrst (
    unsigned short  dbnamel,
    unsigned long   reserved_lgth,
    struct   sqlca  *sqlca,
    _SQLOLDCHAR     *db_alias,
    unsigned long   reset_all,
    _SQLOLDCHAR     *reserved,
    unsigned long   version);
/* ... */
```

API Parameters

dbnamel
Input. A 2-byte unsigned integer representing the length in bytes of the database alias.

reserved_lgth
Reserved for future use. Users should set this value to zero.

sqlca Output. A pointer to the *sqlca* structure.

db_alias
Input. The name that is used to reference the database.

If SQLM_ON is specified for the *reset_all* parameter, this alias is ignored, and the data areas for all active databases are reset.

reset_all
Input. Indicates whether to reset data areas for a specific database, or for all active databases. Set this parameter to one of the following (defined in sqlmon):

SQLM_OFF
Resets data areas for a specific database.

SQLM_ON
Resets data areas for all active databases.

reserved
Reserved for future use. Must be set to NULL.

version
Input. Version ID of the database monitor data to collect. The database monitor only returns data that was available for the requested version. Set this parameter to one of the following symbolic constants:

- SQLM_DBMON_VERSION1

- `SQLM_DBMON_VERSION2`
- `SQLM_DBMON_VERSION5`
- `SQLM_DBMON_VERSION5_2`
- `SQLM_DBMON_VERSION6`

Note: If `SQLM_DBMON_VERSION1` is specified as the version, the APIs cannot be run remotely.

Sample Programs

C	\sqllib\samples\c\db2mon.c

Usage Notes

Each process (attachment) has its own private view of the monitor data. If one user resets, or turns off a monitor switch, other users are not affected. When an application first calls any database monitor function, it inherits the default switch settings from the database manager configuration file . These settings can be overridden with "sqlmon - Get/Update Monitor Switches" on page 355.

If all active databases are reset, some database manager information is also reset to maintain the consistency of the data that is returned.

This API cannot be used to selectively reset specific data items or specific monitor groups. However, a specific group can be reset by turning its switch off, and then on, using "sqlmon - Get/Update Monitor Switches" on page 355.

See Also

"sqlmon - Get/Update Monitor Switches" on page 355

"db2GetSnapshot - Get Snapshot" on page 317

"sqlmonsz - Estimate Size Required for db2GetSnapshot() Output Buffer" on page 358.

UPDATE MONITOR SWITCHES

Turns one or more database monitor recording switches on or off. When the database manager starts, the settings of the six switches are determined by the *dft_mon* database manager configuration parameters .

The database monitor records a base set of information at all times. Users who require more than this basic information can turn on the appropriate switches, but at a cost to system performance. The amount of information available in output from “GET SNAPSHOT” on page 328 reflects which, if any, switches are on.

Authorization

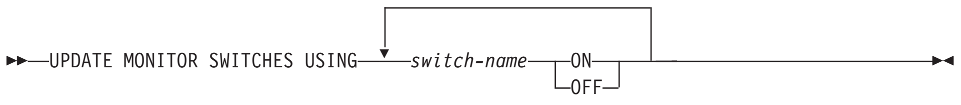
One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

Required Connection

Instance or database:
- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To update the monitor switches at a remote instance (or a different local instance), it is necessary to first attach to that instance.

Command Syntax



Command Parameters

USING switch-name
The following switch names are available:

BUFFERPOOL	Buffer pool activity information
LOCK	Lock information

SORT	Sorting information
STATEMENT	SQL statement information
TABLE	Table activity information
UOW	Unit of work information.

Usage Notes

Information is collected by the database manager only after a switch is turned on. The switches remain set until **db2stop** is issued, or the application that issued the UPDATE MONITOR SWITCHES command terminates. To clear the information related to a particular switch, set the switch off, then on.

Updating switches in one application does not affect other applications.

To view the switch settings, use "GET MONITOR SWITCHES" on page 326.

Appendix B. Logical Data Groupings

The following tables list the logical data groupings and the data elements associated with Snapshot and Event Monitoring.

Table 5. Snapshot Monitor Logical Data Groups and Data Elements

Snapshot Logical Data Groups	**Data Elements**	**See page**
collected	server_db2_type	"Database Manager Type at Monitored (Server) Node" on page 40
	server_version	"Server Version" on page 41
	time_zone_disp	"Time Zone Displacement" on page 44
	time_stamp	"Snapshot Time" on page 249
	node_number	"Node Number" on page 69
	server_prdid	"Server Product / Version ID" on page 41
	server_nname	"Configuration NNAME at Monitoring (Server) Node" on page 39
	server_instance_name	"Server Instance Name" on page 40
	server_switch_list	Monitor Switches Control Data Collected by the Database Manager

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
db2	sort_heap_allocated	"Total Sort Heap Allocated" on page 95
	post_threshold_sorts	"Post Threshold Sorts" on page 96
	piped_sorts_requested	"Piped Sorts Requested" on page 97
	piped_sorts_accepted	"Piped Sorts Accepted" on page 97
	rem_cons_in	"Remote Connections To Database Manager" on page 80
	rem_cons_in_exec	"Remote Connections Executing in the Database Manager" on page 81
	local_cons	"Local Connections" on page 82
	local_cons_in_exec	"Local Connections Executing in the Database Manager" on page 82
	con_local_dbases	"Local Databases with Current Connects" on page 83
	agents_registered	"Agents Registered" on page 86
	agents_waiting_on_token	"Agents Waiting for a Token" on page 86
	db2_status	"Status of Database" on page 48
	agents_registered_top	"Maximum Number of Agents Registered" on page 87
	agents_waiting_top	"Maximum Number of Agents Waiting" on page 87
	comm_private_mem	"Committed Private Memory" on page 91
	idle_agents	"Number of Idle Agents" on page 88
	agents_from_pool	"Agents Assigned From Pool" on page 88
	agents_created_empty_pool	"Agents Created Due to Empty Agent Pool" on page 89
	coord_agents_top	"Maximum Number of Coordinating Agents" on page 90
	max_agent_overflows	"Maximum Agent Overflows" on page 93
	agents_stolen	"Stolen Agents" on page 90
	gw_total_cons	"Total Number of Attempted Connections for DB2 Connect" on page 257
	gw_cur_cons	"Current Number of Connections for DB2 Connect" on page 257
	gw_cons_wait_host	"Number of Connections Waiting for the Host to Reply" on page 258
	gw_cons_wait_client	Number of Connections Waiting for the Client to Send Request
	post_threshold_hash_joins	"Hash Join Threshold" on page 102
	inactive_gw_agents	"Total Inactive DRDA Agents" on page 93
	num_gw_conn_switches	"Connection Switches" on page 94
	db2start_time	"Start Database Manager Timestamp" on page 39
	last_reset	"Last Reset Timestamp" on page 248
	server_switch_list	Monitor Switches Control Data Collected by the Database Manager
	num_nodes_in_db2_instance	"Number of Nodes in Partition" on page 249
	product_name	"Product Name" on page 43
	component_id	"Product Identification" on page 43
	service_level	"Service Level" on page 42
fcm	buff_free	"FCM Buffers Currently Free" on page 105
	buff_free_bottom	"Minimum FCM Buffers Free" on page 105
	MA_free	"Message Anchors Currently Free" on page 106
	MA_free_bottom	"Minimum Message Anchors" on page 106
	CE_free	"Connection Entries Currently Free" on page 106
	CE_free_bottom	"Minimum Connection Entries" on page 107
	RB_free	"Request Blocks Currently Free" on page 107
	RB_free_bottom	"Minimum Request Blocks" on page 108
fcm_node	connection_status	"Connection Status" on page 108
	total_buffers_sent	"Total FCM Buffers Sent" on page 109
	total_buffers_rcvd	"Total FCM Buffers Received" on page 110
	node_number	"Node Number" on page 69

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
dynsql	num_executions	“Statement Executions” on page 235
	num_compilations	“Statement Compilations” on page 235
	prep_time_worst	“Statement Worst Preparation Time” on page 236
	prep_time_best	“Statement Best Preparation Time” on page 236
	int_rows_deleted	“Internal Rows Deleted” on page 194
	int_rows_inserted	“Internal Rows Inserted” on page 196
	rows_read	“Rows Read” on page 193
	int_rows_updated	“Internal Rows Updated” on page 195
	rows_written	“Rows Written” on page 192
	stmt_sorts	“Statement Sorts” on page 224
	total_exec_time	“Elapsed Statement Execution Time” on page 236
	tot_s_cpu_time	“Total System CPU for a Statement” on page 246
	tot_u_cpu_time	“Total User CPU for a Statement” on page 247
	stmt_text	“SQL Dynamic Statement Text” on page 223

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
dbase	pool_data_l_reads	“Buffer Pool Data Logical Reads” on page 113
	pool_data_p_reads	“Buffer Pool Data Physical Reads” on page 115
	pool_data_writes	“Buffer Pool Data Writes” on page 116
	pool_index_l_reads	“Buffer Pool Index Logical Reads” on page 118
	pool_index_p_reads	“Buffer Pool Index Physical Reads” on page 119
	pool_index_writes	“Buffer Pool Index Writes” on page 120
	pool_read_time	“Total Buffer Pool Physical Read Time” on page 122
	pool_write_time	“Total Buffer Pool Physical Write Time” on page 123
	files_closed	“Database Files Closed” on page 124
	pool_async_index_reads	“Buffer Pool Asynchronous Index Reads” on page 128
	pool_data_to_estore	“Buffer Pool Data Pages to Extended Storage” on page 137
	pool_index_to_estore	“Buffer Pool Index Pages to Extended Storage” on page 138
	pool_index_from_estore	“Buffer Pool Index Pages from Extended Storage” on page 140
	pool_data_from_estore	“Buffer Pool Data Pages from Extended Storage” on page 139
	pool_async_data_reads	“Buffer Pool Asynchronous Data Reads” on page 125
	pool_async_data_writes	“Buffer Pool Asynchronous Data Writes” on page 126
	pool_async_index_writes	“Buffer Pool Asynchronous Index Writes” on page 127
	pool_async_read_time	“Buffer Pool Asynchronous Read Time” on page 129
	pool_async_write_time	“Buffer Pool Asynchronous Write Time” on page 130
	pool_async_data_read_reqs	“Buffer Pool Asynchronous Read Requests” on page 131
	direct_reads	“Direct Reads From Database” on page 141
	direct_writes	“Direct Writes to Database” on page 142
	direct_read_reqs	“Direct Read Requests” on page 143
	direct_write_reqs	“Direct Write Requests” on page 144
	direct_read_time	“Direct Read Time” on page 145
	direct_write_time	“Direct Write Time” on page 146
	pool_lsn_gap_clns	“Buffer Pool Log Space Cleaners Triggered” on page 132
	pool_drty_pg_steal_clns	“Buffer Pool Victim Page Cleaners Triggered” on page 133
	pool_drty_pg_thrsh_clns	“Buffer Pool Threshold Cleaners Triggered” on page 134
	locks_held	“Locks Held” on page 164
	lock_waits	“Lock Waits” on page 177
	lock_wait_time	“Time Waited On Locks” on page 178
	lock_list_in_use	“Total Lock List Memory In Use” on page 166
	deadlocks	“Deadlocks Detected” on page 166
	lock_escals	“Number of Lock Escalations” on page 167
	x_lock_escals	“Exclusive Lock Escalations” on page 169
	locks_waiting	“Current Agents Waiting On Locks” on page 179
	sort_heap_allocated	“Total Sort Heap Allocated” on page 95
	total_sorts	“Total Sorts” on page 98
	total_sort_time	“Total Sort Time” on page 99
	sort_overflows	“Sort Overflows” on page 100
	active_sorts	“Active Sorts” on page 101
	commit_sql_stmts	“Commit Statements Attempted” on page 205
	rollback_sql_stmts	“Rollback Statements Attempted” on page 206
	dynamic_sql_stmts	“Dynamic SQL Statements Attempted” on page 203
	static_sql_stmts	“Static SQL Statements Attempted” on page 203

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
dbase (continued)	failed_sql_stmts	"Failed Statement Operations" on page 204
	select_sql_stmts	"Select SQL Statements Executed" on page 207
	ddl_sql_stmts	"Data Definition Language (DDL) SQL Statements" on page 208
	uid_sql_stmts	"Update / Insert / Delete SQL Statements Executed" on page 208
	int_auto_rebinds	"Internal Automatic Rebinds" on page 209
	int_rows_deleted	"Internal Rows Deleted" on page 194
	int_rows_updated	"Internal Rows Updated" on page 195
	int_commits	"Internal Commits" on page 210
	int_rollbacks	"Internal Rollbacks" on page 211
	int_deadlock_rollbacks	"Internal Rollbacks Due To Deadlock" on page 213
	rows_deleted	"Rows Deleted" on page 190
	rows_inserted	"Rows Inserted" on page 190
	rows_updated	"Rows Updated" on page 191
	rows_selected	"Rows Selected" on page 191
	binds_precompiles	"Binds / Precompiles Attempted" on page 214
	total_cons	"Connects Since Database Activation" on page 84
	appls_cur_cons	"Applications Connected Currently" on page 85
	appls_in_db2	"Applications Executing in the Database Currently" on page 85
	sec_log_used_top	"Maximum Secondary Log Space Used" on page 158
	tot_log_used_top	"Maximum Total Log Space Used" on page 159
	sec_logs_allocated	"Secondary Logs Allocated Currently" on page 160
	db_status	"Status of Database" on page 48
	lock_timeouts	"Number of Lock Timeouts" on page 174
	connections_top	"Maximum Number of Concurrent Connections" on page 71
	db_heap_top	"Maximum Database Heap Allocated" on page 156
	int_rows_inserted	"Internal Rows Inserted" on page 196
	log_reads	"Number of Log Pages Read" on page 160
	log_writes	"Number of Log Pages Written" on page 161
	pkg_cache_lookups	"Package Cache Lookups" on page 151
	pkg_cache_inserts	"Package Cache Inserts" on page 153
	cat_cache_lookups	"Catalog Cache Lookups" on page 147
	cat_cache_inserts	"Catalog Cache Inserts" on page 148
	cat_cache_overflows	"Catalog Cache Overflows" on page 148
	cat_cache_heap_full	"Catalog Cache Heap Full" on page 149
	catalog_node	"Catalog Node Number" on page 49
	total_sec_cons	"Secondary Connections" on page 92
	num_assoc_agents	"Number of Associated Agents" on page 92
	agents_top	"Number of Agents Created" on page 238
	coord_agents_top	"Maximum Number of Coordinating Agents" on page 90
	prefetch_wait_time	"Time Waited for Prefetch" on page 135
	appl_section_lookups	"Section Lookups" on page 155
	appl_section_inserts	"Section Inserts" on page 156
	total_hash_joins	"Total Hash Joins" on page 102
	total_hash_loops	"Total Hash Loops" on page 103
	hash_join_overflows	"Hash Join Overflows" on page 103
	hash_join_small_overflows	"Hash Join Small Overflows" on page 104

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
dbase (continued)	pkg_cache_num_overflows	“Package Cache Overflows” on page 153
	pkg_cache_size_top	“Maximum Package Cache Size” on page 154
	total_log_used	“Total Log Space Used” on page 162
	total_log_available	“Total Log Available” on page 163
	db_conn_time	“Database Activation Timestamp” on page 46
	last_reset	“Last Reset Timestamp” on page 248
	last_backup	“Last Backup Timestamp” on page 50
	db_location	“Database Location” on page 49
	server_platform	“Server Operating System” on page 42
	appl_id_oldest_xact	“Application with Oldest Transaction” on page 56
	catalog_node_name	“Catalog Node Network Name” on page 48
	input_db_alias	“Input Database Alias” on page 248
	db_name	“Database Name” on page 45
	db_path	“Database Path” on page 46
rollforward	rf_type	“Rollforward Type” on page 185
	rf_log_num	“Log Being Rolled Forward” on page 185
	rf_status	“Log Phase” on page 186
	rf_timestamp	“Rollforward Timestamp” on page 184
	node_number	“Node Number” on page 69
	ts_name	“Tablespace Being Rolled Forward” on page 185
table_list	last_reset	“Last Reset Timestamp” on page 248
	db_conn_time	“Database Activation Timestamp” on page 46
	input_db_alias	“Input Database Alias” on page 248
	db_name	“Database Name” on page 45
	db_path	“Database Path” on page 46
table	table_file_id	“Table File ID” on page 196
	table_type	“Table Type” on page 187
	rows_written	“Rows Written” on page 192
	rows_read	“Rows Read” on page 193
	overflow_accesses	“Accesses to Overflowed Records” on page 194
	page_reorgs	“Page Reorganizations” on page 197
	table_name	“Table Name” on page 188
	table_schema	“Table Schema Name” on page 189
tablespace_list	last_reset	“Last Reset Timestamp” on page 248
	db_conn_time	“Database Activation Timestamp” on page 46
	input_db_alias	“Input Database Alias” on page 248
	db_name	“Database Name” on page 45
	db_path	“Database Path” on page 46

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
tablespace	pool_data_l_reads	“Buffer Pool Data Logical Reads” on page 113
	pool_data_p_reads	“Buffer Pool Data Physical Reads” on page 115
	pool_async_data_reads	“Buffer Pool Asynchronous Data Reads” on page 125
	pool_data_writes	“Buffer Pool Data Writes” on page 116
	pool_async_data_writes	“Buffer Pool Asynchronous Data Writes” on page 126
	pool_index_l_reads	“Buffer Pool Index Logical Reads” on page 118
	pool_index_p_reads	“Buffer Pool Index Physical Reads” on page 119
	pool_index_writes	“Buffer Pool Index Writes” on page 120
	pool_async_index_writes	“Buffer Pool Asynchronous Index Writes” on page 127
	pool_read_time	“Total Buffer Pool Physical Read Time” on page 122
	pool_write_time	“Total Buffer Pool Physical Write Time” on page 123
	pool_async_read_time	“Buffer Pool Asynchronous Read Time” on page 129
	pool_async_write_time	“Buffer Pool Asynchronous Write Time” on page 130
	pool_async_data_read_reqs	“Buffer Pool Asynchronous Read Requests” on page 131
	direct_reads	“Direct Reads From Database” on page 141
	direct_writes	“Direct Writes to Database” on page 142
	direct_read_reqs	“Direct Read Requests” on page 143
	direct_write_reqs	“Direct Write Requests” on page 144
	direct_read_time	“Direct Read Time” on page 145
	direct_write_time	“Direct Write Time” on page 146
	files_closed	“Database Files Closed” on page 124
	pool_async_index_reads	“Buffer Pool Asynchronous Index Reads” on page 128
	pool_data_to_estore	“Buffer Pool Data Pages to Extended Storage” on page 137
	pool_index_to_estore	“Buffer Pool Index Pages to Extended Storage” on page 138
	pool_index_from_estore	“Buffer Pool Index Pages from Extended Storage” on page 140
	pool_data_from_estore	“Buffer Pool Data Pages from Extended Storage” on page 139
	tablespace_name	“Table Space Name” on page 179
db_lock_list	locks_held	“Locks Held” on page 164
	appls_cur_cons	“Applications Connected Currently” on page 85
	locks_waiting	“Current Agents Waiting On Locks” on page 179
	db_name	“Input Database Alias” on page 248
	db_path	“Database Name” on page 45
		“Database Path” on page 46
appl_lock_list	agent_id	“Application Handle (agent ID)” on page 51
	appl_status	“Application Status” on page 52
	codepage_id	“ID of Code Page Used by Application” on page 55
	locks_held	“Locks Held” on page 164
	locks_waiting	“Current Agents Waiting On Locks” on page 179
	lock_wait_time	“Time Waited On Locks” on page 178
	status_change_time	“Application Status Change Time” on page 55
	appl_id	“Application ID” on page 57
	sequence_no	“Sequence Number” on page 59
	appl_name	“Application Name” on page 56
	auth_id	“Authorization ID” on page 60
	client_db_alias	“Database Alias Used by Application” on page 61

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
lock_wait	subsection_number	“Subsection Number” on page 228
	lock_mode	“Lock Mode” on page 170
	lock_object_type	“Lock Object Type Waited On” on page 172
	agent_id_holding_lk	“Agent ID Holding Lock” on page 181
	lock_mode_requested	“Lock Mode Requested” on page 176
	lock_wait_start_time	“Lock Wait Start Timestamp” on page 180
	lock_escalation	“Lock Escalation” on page 175
	table_name	“Table Name” on page 188
	table_schema	“Table Schema Name” on page 189
	tablespace_name	“Table Space Name” on page 179
	appl_id_holding_lk	“Application ID Holding Lock” on page 182
lock	table_file_id	“Table File ID” on page 196
	lock_object_type	“Lock Object Type Waited On” on page 172
	lock_mode	“Lock Mode” on page 170
	lock_status	“Lock Status” on page 171
	lock_object_name	“Lock Object Name” on page 173
	lock_escalation	“Lock Escalation” on page 175
	table_name	“Table Name” on page 188
	table_schema	“Table Schema Name” on page 189
	tablespace_name	“Table Space Name” on page 179

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
bufferpool	pool_data_l_reads	“Buffer Pool Data Logical Reads” on page 113
	pool_data_p_reads	“Buffer Pool Data Physical Reads” on page 115
	pool_data_writes	“Buffer Pool Data Writes” on page 116
	pool_index_l_reads	“Buffer Pool Index Logical Reads” on page 118
	pool_index_p_reads	“Buffer Pool Index Physical Reads” on page 119
	pool_index_writes	“Buffer Pool Index Writes” on page 120
	pool_read_time	“Total Buffer Pool Physical Read Time” on page 122
	pool_write_time	“Total Buffer Pool Physical Write Time” on page 123
	files_closed	“Database Files Closed” on page 124
	pool_async_data_reads	“Buffer Pool Asynchronous Data Reads” on page 125
	pool_async_data_writes	“Buffer Pool Asynchronous Data Writes” on page 126
	pool_async_index_writes	“Buffer Pool Asynchronous Index Writes” on page 127
	pool_async_read_time	“Buffer Pool Asynchronous Read Time” on page 129
	pool_async_write_time	“Buffer Pool Asynchronous Write Time” on page 130
	pool_async_data_read_reqs	“Buffer Pool Asynchronous Read Requests” on page 131
	direct_reads	“Direct Reads From Database” on page 141
	direct_writes	“Direct Writes to Database” on page 142
	direct_read_reqs	“Direct Read Requests” on page 143
	direct_write_reqs	“Direct Write Requests” on page 144
	direct_read_time	“Direct Read Time” on page 145
	direct_write_time	“Direct Write Time” on page 146
	pool_async_index_reads	“Buffer Pool Asynchronous Index Reads” on page 128
	pool_data_to_estore	“Buffer Pool Data Pages to Extended Storage” on page 137
	pool_index_to_estore	“Buffer Pool Index Pages to Extended Storage” on page 138
	pool_index_from_estore	“Buffer Pool Index Pages from Extended Storage” on page 140
	pool_data_from_estore	“Buffer Pool Data Pages from Extended Storage” on page 139
	bp_name	“Bufferpool Name” on page 135
	input_db_alias	“Input Database Alias” on page 248
	db_name	“Database Name” on page 45
	db_path	“Database Path” on page 46

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
appl_info	agent_id	“Application Handle (agent ID)” on page 51
	appl_status	“Application Status” on page 52
	codepage_id	“ID of Code Page Used by Application” on page 55
	num_assoc_agents	“Number of Associated Agents” on page 92
	coord_node_num	“Coordinating Node” on page 70
	authority_lvl	“Authorization ID” on page 60
	client_pid	“Client Process ID” on page 65
	coord_agent_pid	“Coordinator Agent” on page 78
	status_change_time	“Application Status Change Time” on page 55
	client_platform	“Client Operating Platform” on page 65
	client_protocol	“Client Communication Protocol” on page 66
	country_code	“Database Country Code” on page 67
	appl_name	“Application Name” on page 56
	appl_id	“Application ID” on page 57
	sequence_no	“Sequence Number” on page 59
	auth_id	“Authorization ID” on page 60
	client_nname	“Configuration NNAME of Client” on page 60
	client_prdid	“Client Product/Version ID” on page 61
	input_db_alias	“Input Database Alias” on page 248
	client_db_alias	“Database Alias Used by Application” on page 61
	db_name	“Database Name” on page 45
	db_path	“Database Path” on page 46
	execution_id	“User Login ID” on page 64
	corr_token	“DRDA Correlation Token” on page 64
	outbound_comm_address	“Outbound Communication Address” on page 262
	inbound_comm_address	“Inbound Communication Address” on page 262

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
appl	locks_held	“Locks Held” on page 164
	lock_waits	“Lock Waits” on page 177
	lock_wait_time	“Time Waited On Locks” on page 178
	lock_escals	“Lock Escalation” on page 175
	x_lock_escals	“Exclusive Lock Escalations” on page 169
	deadlocks	“Deadlocks Detected” on page 166
	total_sorts	“Total Sorts” on page 98
	total_sort_time	“Total Sort Time” on page 99
	sort_overflows	“Sort Overflows” on page 100
	pool_data_l_reads	“Buffer Pool Data Logical Reads” on page 113
	pool_data_p_reads	“Buffer Pool Data Physical Reads” on page 115
	pool_data_writes	“Buffer Pool Data Writes” on page 116
	pool_index_l_reads	“Buffer Pool Index Logical Reads” on page 118
	pool_index_p_reads	“Buffer Pool Index Physical Reads” on page 119
	pool_index_writes	“Buffer Pool Index Writes” on page 120
	pool_read_time	“Total Buffer Pool Physical Read Time” on page 122
	pool_write_time	“Total Buffer Pool Physical Write Time” on page 123
	direct_reads	“Direct Reads From Database” on page 141
	direct_writes	“Direct Writes to Database” on page 142
	direct_read_reqs	“Direct Read Requests” on page 143
	direct_write_reqs	“Direct Write Requests” on page 144
	direct_read_time	“Direct Read Time” on page 145
	direct_write_time	“Direct Write Time” on page 146
	pool_data_to_estore	“Buffer Pool Data Pages to Extended Storage” on page 137
	pool_index_to_estore	“Buffer Pool Index Pages to Extended Storage” on page 138
	pool_index_from_estore	“Buffer Pool Index Pages from Extended Storage” on page 140
	pool_data_from_estore	“Buffer Pool Data Pages from Extended Storage” on page 139
	commit_sql_stmts	“Commit Statements Attempted” on page 205
	rollback_sql_stmts	“Rollback Statements Attempted” on page 206
	dynamic_sql_stmts	“Dynamic SQL Statements Attempted” on page 203
	static_sql_stmts	“Static SQL Statements Attempted” on page 203
	failed_sql_stmts	“Failed Statement Operations” on page 204
	select_sql_stmts	“Select SQL Statements Executed” on page 207
	ddl_sql_stmts	“Data Definition Language (DDL) SQL Statements” on page 208
	uid_sql_stmts	“Update/Insert/Delete SQL Statements Executed” on page 208
	int_auto_rebinds	“Internal Automatic Rebinds” on page 209
	int_rows_deleted	“Internal Rows Deleted” on page 194
	int_rows_updated	“Internal Rows Updated” on page 195
	int_commits	“Internal Commits” on page 210
	int_rollbacks	“Internal Rollbacks” on page 211
	int_deadlock_rollbacks	“Internal Rollbacks Due To Deadlock” on page 213
	rows_deleted	“Rows Deleted” on page 190
	rows_inserted	“Rows Inserted” on page 190
	rows_updated	“Rows Updated” on page 191
	rows_selected	“Rows Selected” on page 191
	binds_precompiles	“Binds/Precompiles Attempted” on page 214
	open_rem_curs	“Open Remote Cursors” on page 198
	open_rem_curs_blk	“Open Remote Cursors with Blocking” on page 198
	rej_curs_blk	“Rejected Block Cursor Requests” on page 199

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
appl (continued)	acc_curs_blk	"Accepted Block Cursor Requests" on page 200
	sql_reqs_since_commit	"SQL Requests Since Last Commit" on page 213
	lock_timeouts	"Number of Lock Timeouts" on page 174
	int_rows_inserted	"Internal Rows Inserted" on page 196
	rows_read	"Rows Read" on page 193
	rows_written	"Rows Written" on page 192
	open_loc_curs	"Open Local Cursors" on page 201
	open_loc_curs_blk	"Open Local Cursors with Blocking" on page 201
	pkg_cache_lookups	"Package Cache Lookups" on page 151
	pkg_cache_inserts	"Package Cache Inserts" on page 153
	cat_cache_lookups	"Catalog Cache Lookups" on page 147
	cat_cache_inserts	"Catalog Cache Inserts" on page 148
	cat_cache_overflows	"Catalog Cache Overflows" on page 148
	cat_cache_heap_full	"Catalog Cache Heap Full" on page 149
	num_agents	"Number of Agents Working on a Statement" on page 237
	agents_stolen	"Stolen Agents" on page 90
	associated_agents_top	"Maximum Number of Associated Agents" on page 91
	appl_priority	"Application Agent Priority" on page 67
	appl_priority_type	"Application Priority Type" on page 68
	prefetch_wait_time	"Time Waited for Prefetch" on page 135
	appl_section_lookups	"Section Lookups" on page 155
	appl_section_inserts	"Section Inserts" on page 156
	locks_waiting	"Current Agents Waiting On Locks" on page 179
	total_hash_joins	"Total Hash Joins" on page 102
	total_hash_loops	"Total Hash Loops" on page 103
	hash_join_overflows	"Hash Join Overflows" on page 103
	hash_join_small_overflows	"Hash Join Small Overflows" on page 104
	appl_idle_time	"Application Idle Time" on page 77
	uow_log_space_used	"Unit of Work Log Space Used" on page 162
	uow_lock_wait_time	"Total Time Unit of Work Waited on Locks" on page 180
	uow_comp_status	"Unit of Work Completion Status" on page 75
	agent_usr_cpu_time	"User CPU Time used by Agent" on page 239
	agent_sys_cpu_time	"System CPU Time used by Agent" on page 240
	appl_con_time	"Connection Request Start Timestamp" on page 70
	conn_complete_time	"Connection Request Completion Timestamp" on page 71
	last_reset	"Last Reset Timestamp" on page 248
	uow_start_time	"Unit of Work Start Timestamp" on page 73
	uow_stop_time	"Unit of Work Stop Timestamp" on page 74
	prev_uow_stop_time	"Previous Unit of Work Completion Timestamp" on page 72
	uow_elapsed_time	"Most Recent Unit of Work Elapsed Time" on page 75

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
stmt	num_agents	“Number of Agents Working on a Statement” on page 237
	agents_top	“Number of Agents Created” on page 238
	stmt_type	“Statement Type” on page 216
	stmt_operation	“Statement Operation” on page 217
	section_number	“Section Number” on page 219
	query_cost_estimate	“Query Cost Estimate” on page 227
	query_card_estimate	“Query Number of Rows Estimate” on page 226
	degree_parallelism	“Degree of Parallelism” on page 238
	stmt_sorts	“Statement Sorts” on page 224
	total_sort_time	“Total Sort Time” on page 99
	sort_overflows	“Sort Overflows” on page 100
	rows_read	“Rows Read” on page 193
	rows_written	“Rows Written” on page 192
	int_rows_deleted	“Internal Rows Deleted” on page 194
	int_rows_updated	“Internal Rows Updated” on page 195
	int_rows_inserted	“Internal Rows Inserted” on page 196
	fetch_count	“Number of Successful Fetches” on page 225
	stmt_start	“Statement Operation Start Timestamp” on page 221
	stmt_stop	“Statement Operation Stop Timestamp” on page 221
	stmt_usr_cpu_time	“User CPU Time used by Statement” on page 241
	stmt_sys_cpu_time	“System CPU Time used by Statement” on page 242
	stmt_elapsed_time	“Most Recent Statement Elapsed Time” on page 223
	stmt_node_number	“Statement Node” on page 214
	cursor_name	“Cursor Name” on page 220
	creator	“Application Creator” on page 220
	package_name	“Package Name” on page 218
	stmt_text	“SQL Dynamic Statement Text” on page 223
subsection	ss_exec_time	“Execution Elapsed Time” on page 230
	tq_tot_send_spills	“Total Number of Tablequeue Buffers Overflowed” on page 231
	tq_cur_send_spills	“Current Number of Tablequeue Buffers Overflowed” on page 232
	tq_max_send_spills	“Maximum Number of Tablequeue Buffers Overflows” on page 234
	tq_rows_read	“Number of Rows Read from Tablequeues” on page 233
	tq_rows_written	“Number of Rows Written to Tablequeues” on page 233
	rows_read	“Rows Read” on page 193
	rows_written	“Rows Written” on page 192
	ss_usr_cpu_time	“User CPU Time used by Subsection” on page 245
	ss_sys_cpu_time	“System CPU Time used by Subsection” on page 246
	ss_number	“Subsection Number” on page 228
	ss_status	“Subsection Status” on page 229
	ss_node_number	“Subsection Node Number” on page 229
	tq_node_waited_for	“Waited for Node on a Tablequeue” on page 231
	tq_wait_for_any	“Waiting for Any Node to Send on a Tablequeue” on page 230
	tq_id_waiting_on	“Waited on Node on a Tablequeue” on page 234
agent	agent_pid	“Process or Thread ID” on page 77

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
dcs_dbase	sql_stmts	"Number of SQL Statements Attempted" on page 259
	failed_sql_stmts	"Failed Statement Operations" on page 204
	commit_sql_stmts	"Commit Statements Attempted" on page 205
	rollback_sql_stmts	"Rollback Statements Attempted" on page 206
	rows_selected	"Rows Selected" on page 191
	gw_total_cons	"Total Number of Attempted Connections for DB2 Connect" on page 257
	gw_cur_cons	"Current Number of Connections for DB2 Connect" on page 257
	gw_cons_wait_host	"Number of Connections Waiting for the Host to Reply" on page 258
	gw_cons_wait_client	Number of Connections Waiting for the Client to Send Request
	gw_connections_top	"Maximum Number of Concurrent Connections" on page 256
	gw_comm_errors	"Communication Errors" on page 266
	gw_con_time	"DB2 Connect Gateway First Connect Initiated" on page 256
	outbound_bytes_sent	"Outbound Number of Bytes Sent" on page 263
	outbound_bytes_received	"Inbound Number of Bytes Received" on page 263
	gw_con_time	"DB2 Connect Gateway First Connect Initiated" on page 256
	last_reset	"Last Reset Timestamp" on page 248
	gw_comm_error_time	"Communication Error Time" on page 267
	con_response_time	"Most Recent Response Time for Connect" on page 266
	con_elapsed_time	"Most Recent Connection Elapsed Time" on page 266
	host_response_time	"Host Response Time" on page 265
	dcs_db_name	"DCS Database Name" on page 255
	host_db_name	"Host Database Name" on page 255
dcs_appl_info	agent_id	"Application Handle (agent ID)" on page 51
	codepage_id	"ID of Code Page Used by Application" on page 55
	dcs_appl_status	"DCS Application Status" on page 260
	client_pid	"Client Process ID" on page 65
	status_change_time	"Application Status Change Time" on page 55
	client_platform	"Client Operating Platform" on page 65
	client_protocol	"Client Communication Protocol" on page 66
	host_ccsid	"Host Coded Character Set ID" on page 261
	outbound_comm_protocol	"Outbound Communication Protocol" on page 261
	execution_id	"User Login ID" on page 64
	appl_name	"Application Name" on page 56
	appl_id	"Application ID" on page 57
	sequence_no	"Sequence Number" on page 59
	auth_id	"Authorization ID" on page 60
	client_nname	"Configuration NNAME of Client" on page 60
	client_prdid	"Client Product/Version ID" on page 61
	gw_db_alias	"Database Alias at the Gateway" on page 256
	dcs_db_name	"DCS Database Name" on page 255
	host_db_name	"Host Database Name" on page 255
	host_prdid	"Host Product/Version ID" on page 62
	outbound_appl_id	"Outbound Application ID" on page 62
	outbound_sequence_no	"Outbound Sequence Number" on page 63
	outbound_comm_address	"Outbound Communication Address" on page 262
	inbound_comm_address	"Inbound Communication Address" on page 262

Table 5. Snapshot Monitor Logical Data Groups and Data Elements (continued)

Snapshot Logical Data Groups	Data Elements	See page
dcs_appl	open_cursors	“Number of Open Cursors” on page 259
	appl_idle_time	“Application Idle Time” on page 77
	uow_comp_status	“Unit of Work Completion Status” on page 75
	sql_stmts	“Number of SQL Statements Attempted” on page 259
	failed_sql_stmts	“Failed Statement Operations” on page 204
	commit_sql_stmts	“Commit Statements Attempted” on page 205
	rollback_sql_stmts	“Rollback Statements Attempted” on page 206
	rows_selected	“Rows Selected” on page 191
	inbound_bytes_received	“Inbound Number of Bytes Received” on page 263
	outbound_bytes_sent	“Outbound Number of Bytes Sent” on page 263
	outbound_bytes_received	“Outbound Number of Bytes Received” on page 264
	inbound_bytes_sent	“Inbound Number of Bytes Sent” on page 264
	prev_uow_stop_time	“Previous Unit of Work Completion Timestamp” on page 72
	uow_start_time	“Unit of Work Start Timestamp” on page 73
	uow_stop_time	“Unit of Work Stop Timestamp” on page 74
	last_reset	“Last Reset Timestamp” on page 248
	gw_con_time	“DB2 Connect Gateway First Connect Initiated” on page 256
	gw_exec_time	“Elapsed Time Spent on DB2 Connect Gateway Processing” on page 259
	host_response_time	“Host Response Time” on page 265
	uow_elapsed_time	“Most Recent Unit of Work Elapsed Time” on page 75
	xid	“Transaction ID” on page 265
	tpmon_client_userid	“TP Monitor Client User ID” on page 268
	tpmon_client_wkstn	“TP Monitor Client Workstation Name” on page 268
	tpmon_client_app	“TP Monitor Client Application Name” on page 269
	tpmon_acc_str	“TP Monitor Client Accounting String” on page 269
dcs_stmt	section_number	“Section Number” on page 219
	query_cost_estimate	“Query Cost Estimate” on page 227
	query_card_estimate	“Query Number of Rows Estimate” on page 226
	stmt_operation	“Statement Operation” on page 217
	fetch_count	“Number of Successful Fetches” on page 225
	inbound_bytes_received	“Inbound Number of Bytes Received” on page 263
	outbound_bytes_sent	“Outbound Number of Bytes Sent” on page 263
	outbound_bytes_received	“Outbound Number of Bytes Received” on page 264
	inbound_bytes_sent	“Inbound Number of Bytes Sent” on page 264
	stmt_start	“Statement Operation Start Timestamp” on page 221
	stmt_stop	“Statement Operation Stop Timestamp” on page 221
	gw_exec_time	“Elapsed Time Spent on DB2 Connect Gateway Processing” on page 259
	host_response_time	“Host Response Time” on page 265
	stmt_elpased_time	“Most Recent Statement Elapsed Time” on page 223
	creator	“Application Creator” on page 220
	package_name	“Package Name” on page 218
	stmt_text	“SQL Dynamic Statement Text” on page 223

Table 6. Event Monitor Logical Data Groups and Data Elements

Event Logical Data Groups	Data Elements	See page
db_event	lock_waits	"Lock Waits" on page 177
	lock_wait_time	"Time Waited On Locks" on page 178
	deadlocks	"Deadlocks Detected" on page 166
	lock_escals	"Number of Lock Escalations" on page 167
	lock_escals	"Exclusive Lock Escalations" on page 169
	lock_timeouts	"Number of Lock Timeouts" on page 174
	total_sorts	"Total Sorts" on page 98
	total_sort_time	"Total Sort Time" on page 99
	sort_overflows	"Sort Overflows" on page 100
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 113
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 115
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 125
	pool_data_writes	"Buffer Pool Data Writes" on page 116
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 126
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 118
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 119
	pool_index_writes	"Buffer Pool Index Writes" on page 120
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 127
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 122
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 123
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 129
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 130
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 131
	pool_lsn_gap_clns	"Buffer Pool Log Space Cleaners Triggered" on page 132
	pool_drty_pg_steal_clns	"Buffer Pool Victim Page Cleaners Triggered" on page 133
	pool_drty_pg_thrsh_clns	"Buffer Pool Threshold Cleaners Triggered" on page 134
	direct_reads	"Direct Reads From Database" on page 141
	direct_writes	"Direct Writes to Database" on page 142
	direct_read_reqs	"Direct Read Requests" on page 143
	direct_write_reqs	"Direct Write Requests" on page 144
	direct_read_time	"Direct Read Time" on page 145
	direct_write_time	"Direct Write Time" on page 146
	files_closed	"Database Files Closed" on page 124
	commit_sql_stmts	"Commit Statements Attempted" on page 205
	rollback_sql_stmts	"Rollback Statements Attempted" on page 206
	dynamic_sql_stmts	"Dynamic SQL Statements Attempted" on page 203
	static_sql_stmts	"Static SQL Statements Attempted" on page 203
	failed_sql_stmts	"Failed Statement Operations" on page 204
	select_sql_stmts	"Select SQL Statements Executed" on page 207
	ddl_sql_stmts	"Data Definition Language (DDL) SQL Statements" on page 208
	uid_sql_stmts	"Update/Insert/Delete SQL Statements Executed" on page 208
	int_auto_rebinds	"Internal Automatic Rebinds" on page 209
	int_rows_deleted	"Internal Rows Deleted" on page 194
	int_rows_updated	"Internal Rows Updated" on page 195
	int_rows_inserted	"Internal Rows Inserted" on page 196
	int_commits	"Internal Commits" on page 210
	int_rollbacks	"Internal Rollbacks" on page 211
	rows_deleted	"Rows Deleted" on page 190
	rows_inserted	"Rows Inserted" on page 190
	rows_updated	"Rows Updated" on page 191

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	Data Elements	See page
db_event (continued)	rows_selected	"Rows Selected" on page 191
	binds_precompiles	"Binds / Precompiles Attempted" on page 214
	total_cons	"Connects Since Database Activation" on page 84
	connections_top	"Maximum Number of Concurrent Connections" on page 71
	db_heap_top	"Maximum Database Heap Allocated" on page 156
	sec_log_used_top	"Maximum Secondary Log Space Used" on page 158
	tot_log_used_top	"Maximum Total Log Space Used" on page 159
	log_reads	"Number of Log Pages Read" on page 160
	log_writes	"Number of Log Pages Written" on page 161
	pkg_cache_lookups	"Package Cache Lookups" on page 151
	pkg_cache_inserts	"Package Cache Inserts" on page 153
	cat_cache_lookups	"Catalog Cache Lookups" on page 147
	cat_cache_inserts	"Catalog Cache Inserts" on page 148
	cat_cache_overflows	"Catalog Cache Overflows" on page 148
	cat_cache_heap_full	"Catalog Cache Heap Full" on page 149
	appl_section_lookups	"Section Lookups" on page 155
	appl_section_inserts	"Section Inserts" on page 156
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 128
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 137
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 138
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 140
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 139
	prefetch_wait_time	"Time Waited for Prefetch" on page 135
	catalog_node	"Catalog Node Number" on page 49
	total_hash_joins	"Total Hash Joins" on page 102
	total_hash_loops	"Total Hash Loops" on page 103
	hash_join_overflows	"Hash Join Overflows" on page 103
	hash_join_small_overflows	"Hash Join Small Overflows" on page 104
	pkg_cache_num_overflows	"Package Cache Overflows" on page 153
	pkg_cache_size_top	"Maximum Package Cache Size" on page 154
	disconn_time	"Database Deactivation Timestamp" on page 47
	server_platform	"Server Operating System" on page 42
	talog_node_name	"Catalog Node Network Name" on page 48
	partial_record	"Partial Record" on page 253
dbheader_event	conn_time	"Time of Database Connection" on page 47
	db_name	"Database Name" on page 45
	db_path	"Database Path" on page 46

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	Data Elements	See page
connheader_event	client_pid	"Client Process ID" on page 65
	agent_id	"Application Handle (agent ID)" on page 51
	conn_time	"Time of Database Connection" on page 47
	codepage_id	"ID of Code Page Used by Application" on page 55
	country_code	"Database Country Code" on page 67
	client_platform	"Client Operating Platform" on page 65
	client_protocol	"Client Communication Protocol" on page 66
	node_number	"Node Number" on page 69
	appl_id	"Application ID" on page 57
	sequence_no	"Sequence Number" on page 59
	corr_token	"DRDA Correlation Token" on page 64
	appl_name	"Application Name" on page 56
	auth_id	"Authorization ID" on page 60
	execution_id	"User Login ID" on page 64
	client_nname	"Configuration NNAME of Client" on page 60
	client_prdid	"Client Product/Version ID" on page 61
	client_db_alias	"Database Alias Used by Application" on page 61
start_event	start_time	"Event Start Time" on page 222
deadlock_event	dl_conns	"Connections Involved in Deadlock" on page 175
	rolled_back_agent_id	"Rolled Back Agent" on page 183
	start_time	"Event Start Time" on page 222
	rolled_back_appl_id	"Rolled Back Application" on page 183
	rolled_back_sequence_no	"Rolled Back Sequence Number" on page 184
dlconn_event	lock_mode	"Lock Mode" on page 170
	lock_object_type	"Lock Object Type Waited On" on page 172
	lock_object_name	"Lock Object Name" on page 173
	lock_node	"Lock Node" on page 173
	agent_id	"Application Handle (agent ID)" on page 51
	lock_mode_requested	"Lock Mode Requested" on page 176
	lock_wait_start_time	"Lock Wait Start Timestamp" on page 180
	start_time	"Event Start Time" on page 222
	lock_escalation	"Lock Escalation" on page 175
	appl_id	"Application ID" on page 57
	sequence_no	"Sequence Number" on page 59
	appl_id_holding_lk	"Application ID Holding Lock" on page 182
	sequence_no_holding_lk	"Sequence Number Holding Lock" on page 183
	table_name	"Table Name" on page 188
	table_schema	"Table Schema Name" on page 189
	tablespace_name	"Table Space Name" on page 179
table_event	table_type	"Table Type" on page 187
	rows_written	"Rows Written" on page 192
	rows_read	"Rows Read" on page 193
	overflow_accesses	"Accesses to Overflowed Records" on page 194
	page_reorgs	"Page Reorganizations" on page 197
	event_time	"Event Time" on page 253
	table_name	"Table Name" on page 188
	table_schema	"Table Schema Name" on page 189
	partial_record	"Partial Record" on page 253

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	Data Elements	See page
tablespace_event	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 113
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 115
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 125
	pool_data_writes	"Buffer Pool Data Writes" on page 116
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 126
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 118
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 119
	pool_index_writes	"Buffer Pool Index Writes" on page 120
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 127
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 122
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 123
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 129
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 130
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 131
	direct_reads	"Direct Reads From Database" on page 141
	direct_writes	"Direct Writes to Database" on page 142
	direct_read_reqs	"Direct Read Requests" on page 143
	direct_write_reqs	"Direct Write Requests" on page 144
	direct_read_time	"Direct Read Time" on page 145
	direct_write_time	"Direct Write Time" on page 146
	files_closed	"Database Files Closed" on page 124
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 128
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 137
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 138
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 140
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 139
	event_time	"Event Time" on page 253
	tablespace_name	"Table Space Name" on page 179
	partial_record	"Partial Record" on page 253

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	Data Elements	See page
conn_event	lock_waits	"Lock Waits" on page 177
	lock_wait_time	"Time Waited On Locks" on page 178
	lock_escals	"Lock Escalation" on page 175
	x_lock_escals	"Exclusive Lock Escalations" on page 169
	deadlocks	"Deadlocks Detected" on page 166
	lock_timeouts	"Number of Lock Timeouts" on page 174
	total_sorts	"Total Sorts" on page 98
	total_sort_time	"Total Sort Time" on page 99
	sort_overflows	"Sort Overflows" on page 100
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 113
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 115
	pool_data_writes	"Buffer Pool Data Writes" on page 116
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 118
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 119
	pool_index_writes	"Buffer Pool Index Writes" on page 120
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 122
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 123
	direct_reads	"Direct Reads From Database" on page 141
	direct_writes	"Direct Writes to Database" on page 142
	direct_read_reqs	"Direct Read Requests" on page 143
	direct_write_reqs	"Direct Write Requests" on page 144
	direct_read_time	"Direct Read Time" on page 145
	direct_write_time	"Direct Write Time" on page 146
	commit_sql_stmts	"Commit Statements Attempted" on page 205
	rollback_sql_stmts	"Rollback Statements Attempted" on page 206
	dynamic_sql_stmts	"Dynamic SQL Statements Attempted" on page 203
	static_sql_stmts	"Static SQL Statements Attempted" on page 203
	failed_sql_stmts	"Failed Statement Operations" on page 204
	select_sql_stmts	"Select SQL Statements Executed" on page 207
	ddl_sql_stmts	"Data Definition Language (DDL) SQL Statements" on page 208
	uid_sql_stmts	"Update/Insert/Delete SQL Statements Executed" on page 208
	int_auto_rebinds	"Internal Automatic Rebinds" on page 209
	int_rows_deleted	"Internal Rows Deleted" on page 194
	int_rows_updated	"Internal Rows Updated" on page 195
	int_rows_inserted	"Internal Rows Inserted" on page 196
	int_commits	"Internal Commits" on page 210
	int_rollbacks	"Internal Rollbacks" on page 211
	int_deadlock_rollbacks	"Internal Rollbacks Due To Deadlock" on page 213
	rows_deleted	"Internal Rows Deleted" on page 194
	rows_inserted	"Internal Rows Inserted" on page 196
	rows_updated	"Internal Rows Updated" on page 195
	rows_selected	"Rows Selected" on page 191
	rows_read	"Rows Read" on page 193
	rows_written	"Rows Written" on page 192
	binds_precompiles	"Binds/Precompiles Attempted" on page 214
	rej_curs_blk	"Rejected Block Cursor Requests" on page 199
	acc_curs_blk	"Accepted Block Cursor Requests" on page 200
	pkg_cache_lookups	"Package Cache Lookups" on page 151
	pkg_cache_inserts	"Package Cache Inserts" on page 153
	cat_cache_overflows	"Catalog Cache Overflows" on page 148

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	**Data Elements**	**See page**
conn_event (continued)	cat_cache_heap_full	"Catalog Cache Heap Full" on page 149
	appl_section_lookups	"Section Lookups" on page 155
	appl_section_inserts	"Section Inserts" on page 156
	prefetch_wait_time	"Time Waited for Prefetch" on page 135
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 137
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 138
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 140
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 139
	authority_lvl	"User Authorization Level" on page 68
	coord_node	"Secondary Connections" on page 92
	appl_priority_type	"Application Priority Type" on page 68
	agent_id	"Application Handle (agent ID)" on page 51
	total_hash_joins	"Total Hash Joins" on page 102
	total_hash_loops	"Total Hash Loops" on page 103
	hash_join_overflows	"Hash Join Overflows" on page 103
	hash_join_small_overflows	"Hash Join Small Overflows" on page 104
	cat_cache_inserts	"Catalog Cache Inserts" on page 148
	cat_cache_lookups	"Catalog Cache Lookups" on page 147
	appl_priority	"Application Agent Priority" on page 67
	disconn_time	"Database Deactivation Timestamp" on page 47
	user_cpu_time	"User CPU Time" on page 243
	system_cpu_time	"System CPU Time" on page 244
	appl_id	"Application ID" on page 57
	sequence_no	"Sequence Number" on page 59
	partial_record	"Partial Record" on page 253
sqlca	sqlcode	*Administrative API Reference*
	sqlerrml	*Administrative API Reference*
	sqlcabc	*Administrative API Reference*
	sqlerrmc	*Administrative API Reference*
	sqlerrp	*Administrative API Reference*
	sqlerrd	*Administrative API Reference*
	sqlwarn	*Administrative API Reference*
	sqlstate	*Administrative API Reference*
	sqlcaid	*Administrative API Reference*

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	Data Elements	See page
stmt_event	stmt_type	"Statement Type" on page 216
	stmt_operation	"Statement Operation" on page 217
	fetch_count	"Number of Successful Fetches" on page 225
	section_number	"Section Number" on page 219
	total_sorts	"Total Sorts" on page 98
	total_sort_time	"Total Sort Time" on page 99
	sort_overflows	"Sort Overflows" on page 100
	rows_read	"Rows Read" on page 193
	rows_written	"Rows Written" on page 192
	int_rows_deleted	"Internal Rows Deleted" on page 194
	int_rows_updated	"Internal Rows Updated" on page 195
	int_rows_inserted	"Internal Rows Inserted" on page 196
	agent_id	"Application Handle (agent ID)" on page 51
	agents_top	"Number of Agents Created" on page 238
	start_time	"Event Start Time" on page 222
	stop_time	"Event Stop Time" on page 222
	user_cpu_time	"User CPU Time" on page 243
	system_cpu_time	"System CPU Time" on page 244
	sqlca	"SQL Communications Area (SQLCA)" on page 225
	cursor_name	"Cursor Name" on page 220
	creator	"Application Creator" on page 220
	package_name	"Package Name" on page 218
	appl_id	"Application ID" on page 57
	sequence_no	"Sequence Number" on page 59
	stmt_text	"SQL Dynamic Statement Text" on page 223
	partial_record	"Partial Record" on page 253
subsection_event	agent_id	"Application Handle (agent ID)" on page 51
	ss_exec_time	"Execution Elapsed Time" on page 230
	tq_tot_send_spills	"Total Number of Tablequeue Buffers Overflowed" on page 231
	tq_max_send_spills	Maximum Number of Tablequeue Buffers Overflows
	tq_rows_read	"Number of Rows Read from Tablequeues" on page 233
	tq_rows_written	"Number of Rows Written to Tablequeues" on page 233
	ss_usr_cpu_time	"User CPU Time used by Subsection" on page 245
	ss_sys_cpu_time	"System CPU Time used by Subsection" on page 246
	ss_number	"Subsection Number" on page 228
	ss_node_number	"Subsection Node Number" on page 229
	num_agents	"Number of Agents Working on a Statement" on page 237
	partial_record	"Partial Record" on page 253

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	Data Elements	See page
xaction_event	uow_log_space_used	"Unit of Work Log Space Used" on page 162
	uow_status	"Unit of Work Status" on page 76
	lock_wait_time	"Time Waited On Locks" on page 178
	locks_held_top	"Maximum Number of Locks Held" on page 174
	lock_escals	"Number of Lock Escalations" on page 167
	x_lock_escal	"Exclusive Lock Escalations" on page 169
	rows_written	"Rows Written" on page 192
	agent_id	"Application Handle (agent ID)" on page 51
	user_cpu_time	"User CPU Time" on page 243
	system_cpu_time	"System CPU Time" on page 244
	prev_uow_stop_time	"Previous Unit of Work Completion Timestamp" on page 72
	uow_start_time	"Unit of Work Start Timestamp" on page 73
	stop_time	"Event Stop Time" on page 222
	appl_id	"Application ID" on page 57
	sequence_no	"Sequence Number" on page 59
	partial_record	"Partial Record" on page 253
bufferpool_event	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 113
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 115
	pool_data_writes	"Buffer Pool Data Writes" on page 116
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 118
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 119
	pool_index_writes	"Buffer Pool Index Writes" on page 120
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 122
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 123
	files_closed	"Database Files Closed" on page 124
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 125
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 126
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 127
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 129
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 130
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 131
	direct_reads	"Direct Reads From Database" on page 141
	direct_writes	"Direct Writes to Database" on page 142
	direct_read_reqs	"Direct Read Requests" on page 143
	direct_write_reqs	"Direct Write Requests" on page 144
	direct_read_time	"Direct Read Time" on page 145
	direct_write_time	"Direct Write Time" on page 146
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 128
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 137
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 138
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 140
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 139
	event_time	"Event Time" on page 253
	bp_name	"Bufferpool Name" on page 135
	db_name	"Database Name" on page 45
	db_path	"Database Path" on page 46
	partial_record	"Partial Record" on page 253

Table 6. Event Monitor Logical Data Groups and Data Elements (continued)

Event Logical Data Groups	Data Elements	See page
overflow_event	count	“Number of Event Monitor Overflows” on page 250
	first_overflow_time	“Time of First Event Overflow” on page 250
	last_overflow_time	“Time of Last Event Overflow” on page 251
	node_number	“Node Number” on page 69
log_header_event	byte_order	“Byte Order of Event Data” on page 251
	version	“Version of Monitor Data” on page 252
	num_nodes_in_db2_instance	“Number of Nodes in Partition” on page 249
	codepage_id	“ID of Code Page Used by Application” on page 55
	country_code	“Database Country Code” on page 67
	server_prdid	“Server Product/Version ID” on page 41
	server_instance_name	“Server Instance Name” on page 40
	event_monitor_name	“Event Monitor Name” on page 252

Appendix C. Parallel Edition Version 1.2 Users

In DB2 Version 6 the database system monitor interface has been simplified and is now the same for all database and system configurations. This harmonization of the interface means that some of the request types that were available with the Parallel Edition (PE) V1.2 system monitor are no longer supported.

The most significant change affects how you monitor an application. In DB2 Version 6 an application snapshot returns all the relevaent application information, including a breakdown of the application statistics at the subsection or agent level (if applicable). For example, assuming an application is running a query composed of several subsections, a GET SNAPHOT FOR APPLICATION will return:

- Lock wait information for each agent that is working for this application and is waiting for a lock.
- Tablequeue activity for each subsection executed by this application. This allows you to track progression of a query that is against a partitioned database.
- A list of process IDs or thread IDs for each agent associated with the application.

This information is available on both the coordinator and non-coordinator nodes. In PE V1.2, you would have to request information about individual agents or tablequeues and correlate the output obtained at these levels with the application.

Note: PE V1.2 applications are not compatible with DB2 Version 6.

PE V1.2 applications that are not using any of the requests that have are obsolete in DB2 Version 6 can be recompiled after changing the request type from SQLM_DBMON_PARALLEL1 to SQLM_DBMON_VERSION1. No other changes should be required. See the following tables for obsolete requests.

agent_id

You should note that **agent_id** no longer corresponds to the process ID of the agent process. This field has not been renamed in the API to ensure source compatibility with previous versions, however it has become a globally unique identifier for the application.

Agent ID and application handle are synonymous. See "Partitioned Database Considerations" on page 27 for more information.

API Changes

Obsolete sqlmonss() Request Type	Description	Replacement
SQLMA_AGENT_APPL SQLMA_AGENT_AGENTID	Get snapshot for **agent**	Replaced with SQLMA_APPL which will report a breakdown per agent, if and when applicable.
SQLMA_COORD_AGENTS	List all coordinator agents	Replaced with SQLMA_appl_info_ALL, which returns appl_info for each application. It identifies the node where the coordinator agent runs and provides both its application handle and agent thread or process ID.
SQLMA_FCM_NODE_ALL SQLMA_FCM_NODE	Get Fast Communication Manager	Replaced with SQLMA_DB2, which gets **all** database manager information and returns FCM information (if applicable).
SQLMA_AGENT_ALL SQLMA_COORD_AGENTS	Get snapshot for all agents Get snapshot for coordinator agents	In PE V1.2, returned an SQLMA_AGENT_AGENTID snapshot for all agents, including the coordinators (or the coordinators only). Replaced with SQLMA_APPL_ALL (GET SNAPSHOT FOR APPLICATIONS). Note that information is not returned for agents that are not associated with any applications, as their counts would be zeroes.
SQLMA_DBASE_AGENTS	Get snapshot for all agents for a database	Replaced with SQLMA_DBASE_APPLS.

Obsolete Commands

Obsolete PE V1.2 Command	Replacement
get snapshot for all agents	get snapshot for all applications
get snapshot for all coord agents	get snapshot for all applications
get snapshot for agents on dbname	get snapshot for applications on dbname
get snapshot for agents for application	get snapshot for application
get snapshot for coordinating agent	get snapshot for application
get snapshot for tablequeues	get snapshot for application

Note: GET SNAPSHOT FOR FCM is still supported, however the command processor maps it to a GET SNAPSHOT FOR DBM and extracts the FCM information from the returned output.

Appendix D. DB2 Version 1 sqlestat Users

The following information previously available with the sqlestat API on OS/2 for DB2 Version 1 is now available through snapshot monitoring.

sqlestat Name	**Data Element**
component_id	“Product Identification” on page 43
corr_serv_lvl	“Service Level” on page 42
curr_reqs_lvl	“SQL Requests Since Last Commit” on page 213
db_type	“Server Operating System” on page 42
location	“Database Location” on page 49
node	“Catalog Node Network Name” on page 48
product_name	“Product Name” on page 43

Appendix E. How the DB2 Library Is Structured

The DB2 Universal Database library consists of SmartGuides, online help, books and sample programs in HTML format. This section describes the information that is provided, and how to access it.

To access product information online, you can use the Information Center. You can view task information, DB2 books, troubleshooting information, sample programs, and DB2 information on the Web. See “Accessing Information with the Information Center” on page 406 for details.

Completing Tasks with SmartGuides

SmartGuides help you complete some administration tasks by taking you through each task one step at a time. SmartGuides are available through the Control Center and the Client Configuration Assistant. The following table lists the SmartGuides.

Note: Create Database, Index, and Configure Multisite Update SmartGuide are available for the partitioned database environment.

SmartGuide	Helps You to...	How to Access...
Add Database	Catalog a database on a client workstation.	From the Client Configuration Assistant, click **Add**.
Back up Database	Determine, create, and schedule a backup plan.	From the Control Center, click with the right mouse button on the database you want to back up and select **Backup**->**Database using SmartGuide**.
Configure Multisite Update SmartGuide	Perform a multi-site update, a distributed transaction, or a two-phase commit.	From the Control Center, click with the right mouse button on the **Database** icon and select **Multisite Update**.
Create Database	Create a database, and perform some basic configuration tasks.	From the Control Center, click with the right mouse button on the **Databases** icon and select **Create**->**Database using SmartGuide**.

SmartGuide	Helps You to...	How to Access...
Create Table	Select basic data types, and create a primary key for the table.	From the Control Center, click with the right mouse button on the **Tables** icon and select **Create**->**Table using SmartGuide**.
Create Table Space	Create a new table space.	From the Control Center, click with the right mouse button on the **Table spaces** icon and select **Create**->**Table space using SmartGuide**.
Index	Advise which indexes to create and drop for all your queries.	From the Control Center, click with the right mouse button on the **Index** icon and select **Create**->**Index using SmartGuide**.
Performance Configuration	Tune the performance of a database by updating configuration parameters to match your business requirements.	From the Control Center, click with the right mouse button on the database you want to tune and select **Configure using SmartGuide**.
Restore Database	Recover a database after a failure. It helps you understand which backup to use, and which logs to replay.	From the Control Center, click with the right mouse button on the database you want to restore and select **Restore**->**Database using SmartGuide**.

Accessing Online Help

Online help is available with all DB2 components. The following table describes the various types of help. You can also access DB2 information through the Information Center. For information see “Accessing Information with the Information Center” on page 406.

Type of Help	Contents	How to Access...
Command Help	Explains the syntax of commands in the command line processor.	From the command line processor in interactive mode, enter: ? *command* where *command* is a keyword or the entire command. For example, `? catalog` displays help for all the CATALOG commands, while `? catalog database` displays help for the CATALOG DATABASE command.

Type of Help	Contents	How to Access...
Control Center Help *Client Configuration Assistant Help* *Event Analyzer Help* *Command Center Help*	Explains the tasks you can perform in a window or notebook. The help includes prerequisite information you need to know, and describes how to use the window or notebook controls.	From a window or notebook, click the **Help** push button or press the F1 key.
Message Help	Describes the cause of a message, and any action you should take.	From the command line processor in interactive mode, enter: ? *XXXnnnnn* where *XXXnnnnn* is a valid message identifier. For example, `? SQL30081` displays help about the SQL30081 message. To view message help one screen at a time, enter: ? *XXXnnnnn* | `more` To save message help in a file, enter: ? *XXXnnnnn* > *filename.ext* where *filename.ext* is the file where you want to save the message help.
SQL Help	Explains the syntax of SQL statements.	From the command line processor in interactive mode, enter: `help` *statement* where *statement* is an SQL statement. For example, **help** SELECT displays help about the SELECT statement. **Note:** SQL help is not available on UNIX-based platforms.
SQLSTATE Help	Explains SQL states and class codes.	From the command line processor in interactive mode, enter: **?** *sqlstate* or **?** *class-code* where *sqlstate* is a valid five-digit SQL state and *class-code* is the first two digits of the SQL state. For example, `? 08003` displays help for the 08003 SQL state, while `? 08` displays help for the 08 class code.

DB2 Information – Hardcopy and Online

The table in this section lists the DB2 books. They are divided into two groups:

Cross-platform books
These books contain the common DB2 information for all platforms.

Platform-specific books
These books are for DB2 on a specific platform. For example, there are separate *Quick Beginnings* books for DB2 on OS/2, on Windows NT, and on the UNIX-based platforms.

Cross-platform sample programs in HTML
These samples are the HTML version of the sample programs that are installed with the SDK. They are for informational purposes and do not replace the actual programs.

Most books are available in HTML and PostScript format, or you can choose to order a hardcopy from IBM. The exceptions are noted in the table.

On OS/2 and Windows platforms, HTML documentation files can be installed under the doc\html subdirectory. Depending on the language of your system, some files may be in that language, and the remainder are in English.

On UNIX platforms, you can install multiple language versions of the HTML documentation files under the doc/%L/html subdirectories. Any documentation that is not available in a national language is shown in English.

You can obtain DB2 books and access information in a variety of different ways:

View	See "Viewing Online Information" on page 405.
Search	See "Searching Online Information" on page 408.
Print	See "Printing the PostScript Books" on page 408.
Order	See "Ordering the Printed Books" on page 409.

Name	Description	Form Number File Name for Online Book	HTML Directory
	Cross-Platform Books		

Name	Description	Form Number File Name for Online Book	HTML Directory
Administration Guide	*Administration Guide, Design and Implementation* contains information required to design, implement, and maintain a database. It also describes database access using the Control Center(whether local or in a client/server environment), auditing, database recovery, distributed database support, and high availability. *Administration Guide, Performance* contains information that focuses on the database environment, such as application performance evaluation and tuning. You can order both volumes of the *Administration Guide* in the English language in North America using the form number SBOF-8922.	Volume 1 SC09-2839 db2d1x60 Volume 2 SC09-2840 db2d2x60	db2d0
Administrative API Reference	Describes the DB2 application programming interfaces (APIs) and data structures you can use to manage your databases. Explains how to call APIs from your applications.	SC09-2841 db2b0x60	db2b0
Application Building Guide	Provides environment setup information and step-by-step instructions about how to compile, link, and run DB2 applications on Windows, OS/2, and UNIX-based platforms. This book combines the *Building Applications* books for the OS/2, Windows, and UNIX-based environments.	SC09-2842 db2axx60	db2ax
APPC, CPI-C and SNA Sense Codes	Provides general information about APPC, CPI-C, and SNA sense codes that you may encounter when using DB2 Universal Database products. **Note:** Available in HTML format only.	No form number db2apx60	db2ap

Name	Description	Form Number File Name for Online Book	HTML Directory
Application Development Guide	Explains how to develop applications that access DB2 databases using embedded SQL or JDBC, how to write stored procedures, user-defined types, user-defined functions, and how to use triggers. It also discusses programming techniques and performance considerations. This book was formerly known as the *Embedded SQL Programming Guide*.	SC09-2845 db2a0x60	db2a0
CLI Guide and Reference	Explains how to develop applications that access DB2 databases using the DB2 Call Level Interface, a callable SQL interface that is compatible with the Microsoft ODBC specification.	SC09-2843 db2l0x60	db2l0
Command Reference	Explains how to use the command line processor, and describes the DB2 commands you can use to manage your database.	SC09-2844 db2n0x60	db2n0
Data Movement Utilities Guide and Reference	Explains how to use the Load, Import, Export, Autoloader, and Data Propogation utilities to work with the data in the database.	SC09-2858 db2dmx60	db2dm
DB2 Connect Personal Edition Quick Beginnings	Provides planning, installing, and configuring information for DB2 Connect Personal Edition.	GC09-2830 db2c1x60	db2c1
DB2 Connect User's Guide	Provides concepts, programming and general usage information about the DB2 Connect products.	SC09-2838 db2c0x60	db2c0
Connectivity Supplement	Provides setup and reference information on how to use DB2 for AS/400, DB2 for OS/390, DB2 for MVS, or DB2 for VM as DRDA application requesters with DB2 Universal Database servers, and on how to use DRDA application servers with DB2 Connect application requesters. **Note:** Available in HTML and PostScript formats only.	No form number db2h1x60	db2h1
Glossary	Provides a comprehensive list of all DB2 terms and definitions. **Note:** Available in HTML format only.	No form number db2t0x50	db2t0

Name	Description	Form Number File Name for Online Book	HTML Directory
Installation and Configuration Supplement	Guides you through the planning, installation, and set up of platform-specific DB2 clients. This supplement contains information on binding, setting up client and server communications, DB2 GUI tools, DRDA AS, distributed installation, and the configuration of distributed requests and access methods to heterogeneous data sources.	GC09-2857 db2iyx60	db2iy
Message Reference	Lists messages and codes issued by DB2, and describes the actions you should take.	GC09-2846 db2m0x60	db2m0
Replication Guide and Reference	Provides planning, configuration, administration, and usage information for the IBM Replication tools supplied with DB2.	SC26-9642 db2e0x60	db2e0
SQL Getting Started	Introduces SQL concepts, and provides examples for many constructs and tasks.	SC09-2856 db2y0x60	db2y0
SQL Reference, Volume 1 and Volume 2	Describes SQL syntax, semantics, and the rules of the language. Also includes information about release-to-release incompatibilities, product limits, and catalog views. You can order both volumes of the *SQL Reference* in the English language in North America with the form number SBOF-8923.	SBOF-8923 Volume 1 db2s1x60 Volume 2 db2s2x60	db2s0
System Monitor Guide and Reference	Describes how to collect different kinds of information about databases and the database manager. Explains how to use the information to understand database activity, improve performance, and determine the cause of problems.	SC09-2849 db2f0x60	db2f0
Troubleshooting Guide	Helps you determine the source of errors, recover from problems, and use diagnostic tools in consultation with DB2 Customer Service.	S10J-8169	db2p0

Name	Description	Form Number File Name for Online Book	HTML Directory
What's New	Describes the new features, functions, and enhancements in DB2 Universal Database, Version 6.0, including information about Java-based tools.	SC09-2851 db2q0x60	db2q0
	Platform-Specific Books		
Administering Satellites Guide and Reference	Provides planning, configuration, administration, and usage information for satellites.	GC09-2821 db2dsx60	db2ds
DB2 Personal Edition Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database Personal Edition on the OS/2, Windows 95, and Windows NT operating systems.	GC09-2831 db2i1x60	db2i1
DB2 for OS/2 Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database on the OS/2 operating system. Also contains installing and setup information for many supported clients.	GC09-2834 db2i2x60	db2i2
DB2 for UNIX Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database on UNIX-based platforms. Also contains installing and setup information for many supported clients.	GC09-2836 db2ixx60	db2ix
DB2 for Windows NT Quick Beginnings	Provides planning, installation, migration, and configuration information for DB2 Universal Database on the Windows NT operating system. Also contains installing and setup information for many supported clients.	GC09-2835 db2i6x60	db2i6
DB2 Enterprise - Extended Edition for UNIX Quick Beginnings	Provides planning, installation, and configuration information for DB2 Enterprise - Extended Edition for UNIX. Also contains installing and setup information for many supported clients.	GC09-2832 db2v3x60	db2v3

Name	**Description**	**Form Number** **File Name for Online Book**	**HTML Directory**
DB2 Enterprise - Extended Edition for Windows NT Quick Beginnings	Provides planning, installation, and configuration information for DB2 Enterprise - Extended Edition for Windows NT. Also contains installing and setup information for many supported clients.	GC09-2833 db2v6x60	db2v6
DB2 Connect Enterprise Edition for OS/2 and Windows NT Quick Beginnings	Provides planning, migration, installation, and configuration information for DB2 Connect Enterprise Edition on the OS/2 and Windows NT operating systems. Also contains installation and setup information for many supported clients. This book was formerly part of the *DB2 Connect Enterprise Edition Quick Beginnings*.	GC09-2828 db2c6x60	db2c6
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	Provides planning, migration, installation, configuration, and usage information for DB2 Connect Enterprise Edition in UNIX-based platforms. Also contains installation and setup information for many supported clients. This book was formerly part of the *DB2 Connect Enterprise Edition Quick Beginnings*.	GC09-2829 db2cyx60	db2cy
DB2 Data Links Manager for AIX Quick Beginnings	Provides planning, installation, configuration, and task information for DB2 Data Links Manager for AIX.	GC09-2837 db2z0x60	db2z0
DB2 Data Links Manager for Windows NT Quick Beginnings	Provides planning, installation, configuration, and task information for DB2 Data Links Manager for Windows NT.	GC09-2827 db2z6x60	db2z6
DB2 Query Patroller Administration Guide	Provides administration information on DB2 Query Patrol.	SC09-2859 db2dwx60	db2dw
DB2 Query Patroller Installation Guide	Provides installation information on DB2 Query Patrol.	GC09-2860 db2iwx60	db2iw
DB2 Query Patroller User's Guide	Describes how to use the tools and functions of the DB2 Query Patrol.	SC09-2861 db2wwx60	db2ww

Name	Description	Form Number File Name for Online Book	HTML Directory
	Cross-Platform Sample Programs in HTML		
Sample programs in HTML	Provides the sample programs in HTML format for the programming languages on all platforms supported by DB2 for informational purposes (not all samples are available in all languages). Only available when the SDK is installed. See *Application Building Guide* for more information on the actual programs. **Note:** Available in HTML format only.	No form number	db2hs/c db2hs/cli db2hs/clp db2hs/cpp db2hs/cobol db2hs/cobol_mf db2hs/fortran db2hs/java db2hs/rexx

Notes:

1. The character in the sixth position of the file name indicates the language of a book. For example, the file name `db2d0e60` indicates that the *Administration Guide* is in English. The following letters are used in the file names to indicate the language of a book:

Language	Identifier
Brazilian Portuguese	b
Bulgarian	u
Czech	x
Danish	d
Dutch	q
English	e
Finnish	y
French	f
German	g
Greek	a
Hungarian	h
Italian	i
Japanese	j
Korean	k
Norwegian	n
Polish	p
Portuguese	v
Russian	r
Simp. Chinese	c
Slovenian	l
Spanish	z

Swedish	s
Trad. Chinese	t
Turkish	m

2. For late breaking information that could not be included in the DB2 books:
   - On UNIX-based platforms, see the Release.Notes file. This file is located in the DB2DIR/Readme/%L directory, where `%L` is the locale name and DB2DIR is:
     - /usr/lpp/db2_06_01 on AIX
     - /opt/IBMdb2/V6.1 on HP-UX, Solaris, SCO UnixWare 7, and Silicon Graphics IRIX
     - /usr/IBMdb2/V6.1 on Linux.
   - On other platforms, see the RELEASE.TXT file. This file is located in the directory where the product is installed.
   - Under Windows `Start` menu

Viewing Online Information

The manuals included with this product are in Hypertext Markup Language (HTML) softcopy format. Softcopy format enables you to search or browse the information, and provides hypertext links to related information. It also makes it easier to share the library across your site.

You can view the online books or sample programs with any browser that conforms to HTML Version 3.2 specifications.

To view online books or sample programs on all platforms other than SCO UnixWare 7:

- If you are running DB2 administration tools, use the Information Center. See “Accessing Information with the Information Center” on page 406 for details.
- Select the `Open Page` menu item of your Web browser. The page you open contains descriptions of and links to DB2 information:
  - On UNIX-based platforms, open the following page:

    ```
    file:/INSTHOME/sqllib/doc/%L/html/index.htm
    ```

    where *%L* is the locale name.
  - On other platforms, open the following page:

    ```
    sqllib\doc\html\index.htm
    ```

    The path is located on the drive where DB2 is installed.

If you have not installed the Information Center, you can open the page by double-clicking on the **DB2 Online Books** icon. Depending on the system you are using, the icon is in the main product folder or the Windows Start menu.

To view online books or sample programs on the SCO UnixWare 7:

- DB2 Universal Database for SCO UnixWare 7 uses the native SCOhelp utility to search the DB2 information. You can access SCOhelp by the following methods:
  - entering the ″scohelp″ command on the command line,
  - selecting the `Help` menu in the Control Panel of the CDE desktop or
  - selecting `Help` in the Root menu of the Panorama desktop

  For more information on SCOhelp, refer to the *Installation and Configuration Supplement*.

Accessing Information with the Information Center

The Information Center provides quick access to DB2 product information. The Information Center is available on all platforms on which the DB2 administration tools are available.

Depending on your system, you can access the Information Center from the:

- Main product folder
- Toolbar in the Control Center
- Windows Start menu
- Help menu of the Control Center

The Information Center provides the following kinds of information. Click the appropriate tab to look at the information:

Tasks	Lists tasks you can perform using DB2.
Reference	Lists DB2 reference information, such as keywords, commands, and APIs.
Books	Lists DB2 books.
Troubleshooting	Lists categories of error messages and their recovery actions.
Sample Programs	Lists sample programs that come with the DB2 Software Developer’s Kit. If the Software Developer’s Kit is not installed, this tab is not displayed.
Web	Lists DB2 information on the World Wide

Web. To access this information, you must have a connection to the Web from your system.

When you select an item in one of the lists, the Information Center launches a viewer to display the information. The viewer might be the system help viewer, an editor, or a Web browser, depending on the kind of information you select.

The Information Center provides some search capabilities, so you can look for specific topics, and filter capabilities to limit the scope of your searches.

For a full text search, click the `Search` button of the Information Center follow the *Search DB2 Books* link in each HTML file.

The HTML search server is usually started automatically. If a search in the HTML information does not work, you may have to start the search server by double-clicking its icon on the Windows or OS/2 desktop.

Refer to the release notes if you experience any other problems when searching the HTML information.

Note: Search function is not available in the Linux and Silicon Graphics environments.

Setting Up a Document Server

By default, the DB2 information is installed on your local system. This means that each person who needs access to the DB2 information must install the same files. To have the DB2 information stored in a single location, use the following instructions:

1. Copy all files and subdirectories from \sqllib\doc\html on your local system to a Web server. Each book has its own subdirectory containing all the necessary HTML and GIF files that make up the book. Ensure that the directory structure remains the same.
2. Configure the Web server to look for the files in the new location. For information, see the NetQuestion Appendix in *Installation and Configuration Supplement*.
3. If you are using the Java version of the Information Center, you can specify a base URL for all HTML files. You should use the URL for the list of books.
4. Once you are able to view the book files, you should bookmark commonly viewed topics. Among those, you will probably want to bookmark the following pages:

- List of books
- Tables of contents of frequently used books
- Frequently referenced articles, such as the *ALTER TABLE* topic
- The Search form

For information about setting up a search, see the NetQuestion Appendix in *Installation and Configuration Supplement* book.

Searching Online Information

To search for information in the HTML books, you can do the following:

- Click on **Search the DB2 Books** at the bottom of any page in the HTML books. Use the search form to find a specific topic. This function is not available in the Linux or Silicon Graphics IRIX environments.
- Click on **Index** at the bottom of any page in an HTML book. Use the index to find a specific topic in the book.
- Display the table of contents or index of the HTML book, and then use the find function of the Web browser to find a specific topic in the book.
- Use the bookmark function of the Web browser to quickly return to a specific topic.
- Use the search function of the Information Center to find specific topics. See "Accessing Information with the Information Center" on page 406 for details.

Printing the PostScript Books

If you prefer to have printed copies of the manuals, you can decompress and print PostScript versions. For the file name of each book in the library, see the table in "DB2 Information – Hardcopy and Online" on page 398. Specify the full path name for the file you intend to print.

On OS/2 and Windows platforms:

1. Copy the compressed PostScript files to a hard drive on your system. The files have a file extension of .exe and are located in the x:\doc\language\books\ps directory, where x: is the letter representing the CD-ROM drive and *language* is the two-character country code that represents your language (for example, EN for English).
2. Decompress the file that corresponds to the book that you want. Each compressed book is a self-extracting executable file. To decompress the

book, simply run it as you would run any other executable program. The result from this step is a printable PostScript file with a file extension of .ps.

3. Ensure that your default printer is a PostScript printer capable of printing Level 1 (or equivalent) files.
4. Enter the following command from a command line:
   ```
   print filename.ps
   ```

On UNIX-based platforms:

1. Mount the CD-ROM. Refer to your *Quick Beginnings* manual for the procedures to mount the CD-ROM.
2. Change to /cdrom/doc/%L/ps directory on the CD-ROM, where */cdrom* is the mount point of the CD-ROM and *%L* is the name of the desired locale. The manuals will be installed in the previously-mentioned directory with file names ending with .ps.Z.
3. Decompress and print the manual you require using the following command:
   - For AIX:
     ```
     zcat filename | qprt -P PSPrinter_queue
     ```
   - For HP-UX, Solaris, or SCO UnixWare 7:
     ```
     zcat filename | lp -d PSPrinter_queue
     ```
   - For Linux:
     ```
     zcat filename | lpr -P PSPrinter_queue
     ```
   - For Silicon Graphics IRIX:
     ```
     zcat < filename | lp -d PSPrinter_queue
     ```

   where *filename* is the full path name and extension of the compressed PostScript file and *PSprinter_queue* is the name of the PostScript printer queue.

   For example, to print the English version of *DB2 for UNIX Quick Beginnings* on AIX, you can use the following command:
   ```
   zcat /cdrom/doc/en/ps/db2ixe60.ps.Z || qprt -P ps1
   ```

Ordering the Printed Books

You can order the printed DB2 manuals either as a set or individually. There are three sets of books available. The form number for the entire set of DB2 books is SBOF-8926-00. The form number for the books listed under the heading "Cross-Platform Books" is SBOF-8924-00.

Note: These form numbers only apply if you are ordering books that are printed in the English language in North America.

You can also order books individually by the form number listed in “DB2 Information – Hardcopy and Online” on page 398. To order printed versions, contact your IBM authorized dealer or marketing representative, or phone `1-800-879-2755` in the United States or `1-800-IBM-4YOU` in Canada.

Appendix F. Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the

IBM Director of Licensing
IBM Corporation, North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

This publication may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and/or other countries:

ACF/VTAM	MVS/ESA
ADSTAR	MVS/XA
AISPO	OS/400
AIX	OS/390
AIXwindows	OS/2
AnyNet	PowerPC
APPN	QMF
AS/400	RACF
CICS	RISC System/6000
C Set++	SP
C/370	SQL/DS
DATABASE 2	SQL/400
DataHub	S/370
DataJoiner	System/370
DataPropagator	System/390
DataRefresher	SystemView
DB2	VisualAge
DB2 Connect	VM/ESA
DB2 Universal Database	VSE/ESA
Distributed Relational Database Architecture	VTAM
DRDA	WIN-OS/2
Extended Services	
FFST	
First Failure Support Technology	
IBM	
IMS	
LAN Distance	

Trademarks of Other Companies

The following terms are trademarks or registered trademarks of the companies listed:

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

HP-UX is a trademark of Hewlett-Packard.

Java, HotJava, Solaris, Solstice, and Sun are trademarks of Sun Microsystems, Inc.

Index

Contacting IBM

This section lists ways you can get more information from IBM.

If you have a technical problem, please take the time to review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. Depending on the nature of your problem or concern, this guide will suggest information you can gather to help us to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

Telephone

If you live in the U.S.A., call one of the following numbers:
- 1-800-237-5511 to learn about available service options.
- 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, see Appendix A of the IBM Software Support Handbook. You can access this document by accessing the following page:

```
http://www.ibm.com/support/
```

then performing a search using the keyword “handbook”.

Note that in some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

World Wide Web
http://www.software.ibm.com/data/
http://www.software.ibm.com/data/db2/library/

The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more. The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information. (Note that this information may be in English only.)

Anonymous FTP Sites
ftp.software.ibm.com

Log on as anonymous. In the directory /ps/products/db2, you can find demos, fixes, information, and tools concerning DB2 and many related products.

Internet Newsgroups
comp.databases.ibm-db2, bit.listserv.db2-l

These newsgroups are available for users to discuss their experiences with DB2 products.

CompuServe
GO IBMDB2 to access the IBM DB2 Family forums

All DB2 products are supported through these forums.

To find out about the IBM Professional Certification Program for DB2 Universal Database, go to http://www.software.ibm.com/data/db2/db2tech/db2cert.html

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC09-2849-00